CASE stands for **C**omputer **A**ided **S**oftware **E**ngineering. It means, development and maintenance of software projects with help of various automated software tools.

CASE Tools

CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.

There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.

Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

Components of CASE Tools

CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:

- **Central Repository** - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.
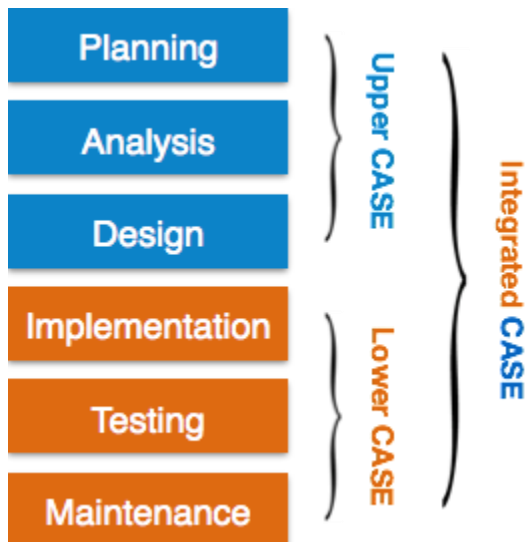
- **Upper Case Tools** - Upper CASE tools are used in planning, analysis and design stages of SDLC.

- **Lower Case Tools** - Lower CASE tools are used in implementation, testing and maintenance.

- **Integrated Case Tools** - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

CASE tools can be grouped together if they have similar functionality, process activities and capability of getting integrated with other tools.

Scope of Case Tools

The scope of CASE tools goes throughout the SDLC.

Case Tools Types

Now we briefly go through various CASE tools

Diagram tools

These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.

Process Modeling Tools

Process modeling is method to create software process model, which is used to develop the software. Process modeling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF Composer

Project Management Tools

These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

Documentation Tools

Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project.

Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.

Analysis Tools

These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.

Design Tools

These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design

Configuration Management Tools

An instance of software is released under one version. Configuration Management tools deal with –

- Version and revision management
- Baseline configuration management
- Change control management

CASE tools help in this by automatic tracking, version management and release management. For example, Fossil, Git, Accu REV.

Change Control Tools

These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released. CASE tools automate change tracking, file management, code management and more. It also helps in enforcing change policy of the organization.

Programming Tools

These tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse.

Prototyping Tools

Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product.

Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information. In addition, they provide simulation of software prototype. For example, Serena prototype composer, Mockup Builder.

Web Development Tools

These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect, Foundation 3, Brackets.

Quality Assurance Tools

Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. For example, SoapTest, AppsWatch, JMeter.

Maintenance Tools

Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP Quality Center.

What is Project?

A project is a group of tasks that need to complete to reach a clear result. A project also defines as a set of inputs and outputs which are required to achieve a goal. Projects can vary from simple to difficult and can be operated by one person or a hundred.

Projects usually described and approved by a project manager or team executive. They go beyond their expectations and objects, and it's up to the team to handle logistics and complete the project on time. For good project development, some teams split the project into specific tasks so they can manage responsibility and utilize team strengths.

What is software project management?

Software project management is an art and discipline of planning and supervising software projects. It is a sub-discipline of software project management in which software projects planned, implemented, monitored and controlled.

It is a procedure of managing, allocating and timing resources to develop computer software that fulfills requirements.

In software Project Management, the client and the developers need to know the length, period and cost of the project.

Prerequisite of software project management?

There are three needs for software project management. These are:

1. Time
2. Cost
3. Quality

It is an essential part of the software organization to deliver a quality product, keeping the cost within the client?s budget and deliver the project as per schedule. There are various factors, both external and internal, which may impact this triple factor. Any of three-factor can severely affect the other two.

Project Manager

A project manager is a character who has the overall responsibility for the planning, design, execution, monitoring, controlling and closure of a project. A project manager represents an essential role in the achievement of the projects.

A project manager is a character who is responsible for giving decisions, both large and small projects. The project manager is used to manage the risk and minimize uncertainty. Every decision the project manager makes must directly profit their project.

Role of a Project Manager:

1. Leader

A project manager must lead his team and should provide them direction to make them understand what is expected from all of them.

2. Medium:

The Project manager is a medium between his clients and his team. He must coordinate and transfer all the appropriate information from the clients to his team and report to the senior management.

3. Mentor:

He should be there to guide his team at each step and make sure that the team has an attachment. He provides a recommendation to his team and points them in the right direction.

Responsibilities of a Project Manager:
1. Managing risks and issues.
2. Create the project team and assigns tasks to several team members.
3. Activity planning and sequencing.
4. Monitoring and reporting progress.
5. Modifies the project plan to deal with the situation.

Activities

Software Project Management consists of many activities, that includes planning of the project, deciding the scope of product, estimation of cost in different terms, scheduling of tasks, etc.

The list of activities are as follows:

1. Project planning and Tracking
2. Project Resource Management
3. Scope Management
4. Estimation Management
5. Project Risk Management
6. Scheduling Management
7. Project Communication Management
8. Configuration Management

Now we will discuss all these activities -

1. Project Planning: It is a set of multiple processes, or we can say that it a task that performed before the construction of the product starts.

2. Scope Management: It describes the scope of the project. Scope management is important because it clearly defines what would do and what would not. Scope Management create the project to contain restricted and quantitative tasks, which may merely be documented and successively avoids price and time overrun.

3. Estimation management: This is not only about cost estimation because whenever we start to develop software, but we also figure out their size(line of code), efforts, time as well as cost.

If we talk about the size, then Line of code depends upon user or software requirement.

If we talk about effort, we should know about the size of the software, because based on the size we can quickly estimate how big team required to produce the software.

If we talk about time, when size and efforts are estimated, the time required to develop the software can easily determine.

And if we talk about cost, it includes all the elements such as:

- o Size of software
- o Quality
- o Hardware
- o Communication
- o Training
- o Additional Software and tools
- o Skilled manpower

4. Scheduling Management: Scheduling Management in software refers to all the activities to complete in the specified order and within time slotted to each activity. Project managers define multiple tasks and arrange them keeping various factors in mind.

For scheduling, it is compulsory -

- o Find out multiple tasks and correlate them.
- o Divide time into units.
- o Assign the respective number of work-units for every job.
- o Calculate the total time from start to finish.
- o Break down the project into modules.

5. Project Resource Management: In software Development, all the elements are referred to as resources for the project. It can be a human resource, productive tools, and libraries.

Resource management includes:

- o Create a project team and assign responsibilities to every team member
- o Developing a resource plan is derived from the project plan.

o   Adjustment of resources.

6. Project Risk Management: Risk management consists of all the activities like identification, analyzing and preparing the plan for predictable and unpredictable risk in the project.

Several points show the risks in the project:

o   The Experienced team leaves the project, and the new team joins it.
o   Changes in requirement.
o   Change in technologies and the environment.
o   Market competition.

7. Project Communication Management: Communication is an essential factor in the success of the project. It is a bridge between client, organization, team members and as well as other stakeholders of the project such as hardware suppliers.

From the planning to closure, communication plays a vital role. In all the phases, communication must be clear and understood. Miscommunication can create a big blunder in the project.

8. Project Configuration Management: Configuration management is about to control the changes in software like requirements, design, and development of the product.

The Primary goal is to increase productivity with fewer errors.

Some reasons show the need for configuration management:

o   Several people work on software that is continually update.
o   Help to build coordination among suppliers.
o   Changes in requirement, budget, schedule need to accommodate.
o   Software should run on multiple systems.

Tasks perform in Configuration management:

o   Identification
o   Baseline
o   Change Control
o   Configuration Status Accounting
o   Configuration Audits and Reviews

People involved in Configuration Management:

Project Management Tools

To manage the Project management system adequately and efficiently, we use Project management tools.

Here are some standard tools:

Gantt chart

Henry Gantt developed the Gantt Chart in 1917. Gantt chart is a handy tool when you want to see the whole landscape of either one or multiple projects. It helps you to view which functions are dependent on one another and which event is coming up.

Gantt charts are primarily used to allocate resources to activities. The funds allocated to activities include staff, hardware, and software. Gantt charts are useful for resource planning. A Gantt chart is a particular type of bar chart where each bar represents an activity. The bars are drawn along a timeline. The distance of each bar is proportional to the duration of time planned for the corresponding event. Gantt charts are used in software project management are an enlarged version of the regular Gantt charts. In the Gantt charts used for software project management, each bar subsists of a white part and a shaded section. The shaded part of the bar displays the length of time every task is estimated to take. The white part displays the slack time, that is, the current time by which a method must be completed. A Gantt chart representation for the MIS problem, as shown in fig:

Gantt chart representation of the MIS Problem
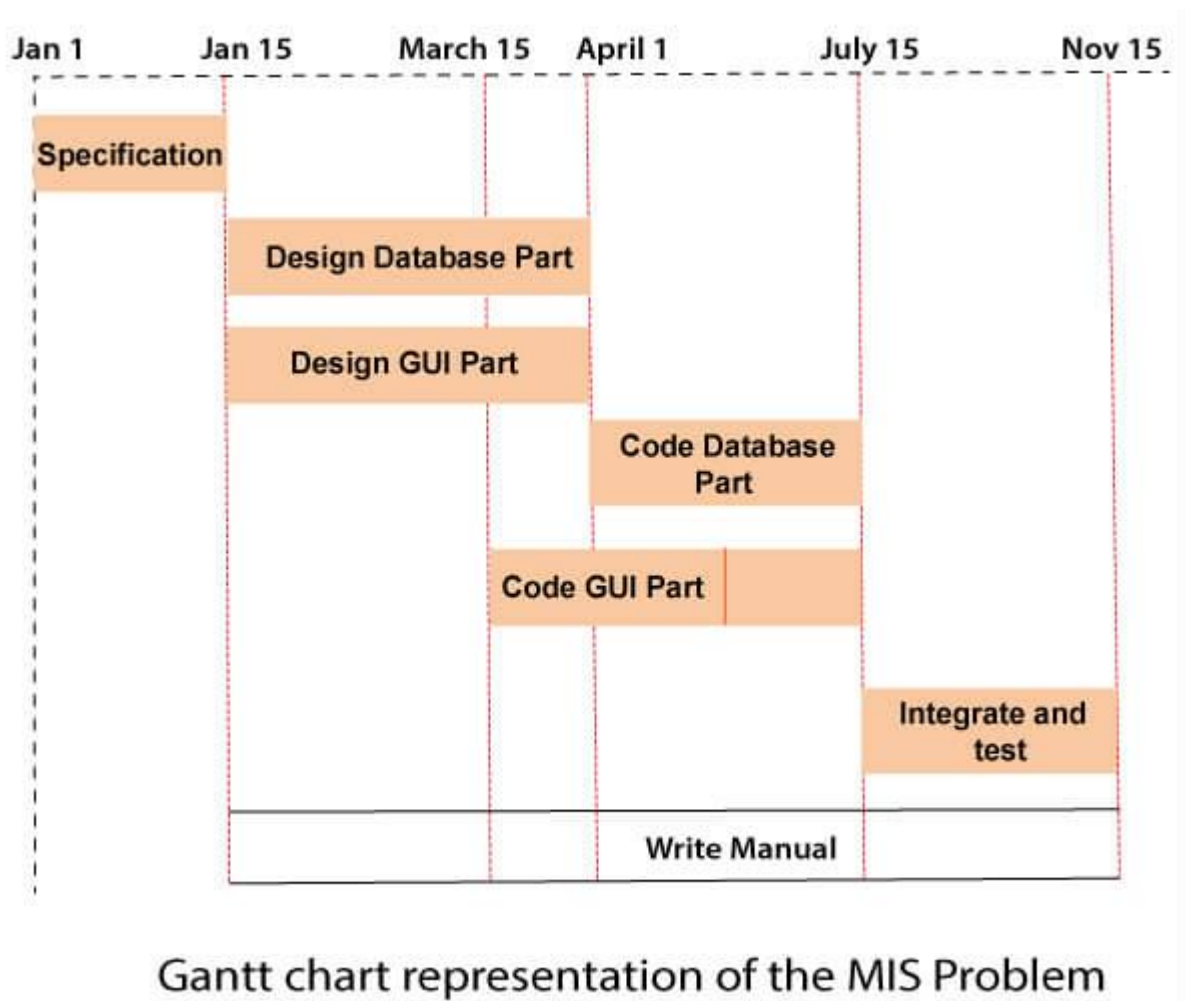
PERT chart

PERT (Project Evaluation and Review Technique) charts contain a network of boxes and arrows. The boxes show activities, and the arrows represent function dependencies. PERT chart represents the numerical variations in the plan estimates assuming a normal distribution. Thus, in a PERT chart consist of making a single estimate for each function, pessimistic, likely, and optimistic size is made. The boxes of PERT charts are generally annotated with the pessimistic, likely, and optimistic estimates for each method. Since all possible completion times between the minimum and maximum period for each process has to be treated, there are not one but many critical ways, depending on the permutations of the estimates for each purpose. This makes the analytical path method in PERT charts very complicated. A critical way in a PERT chart is shown by utilizing thicker arrows. The PERT chart representation of the MIS problem as shown in fig. PERT charts are a more sophisticated method of activity chart. Inactivity diagrams only the estimated method durations are represented. Since the actual time might vary from the estimated time, the utility of the activity diagrams is finite.

Gantt chart representation of a project record is useful in planning the usage of resources, while PERT chart is useful for monitoring the proper progress of activities. Also, it is easier to identify parallel operations in a project utilizing a PERT chart. Project managers use to determine the parallel activities in a project for assignment to various engineers.
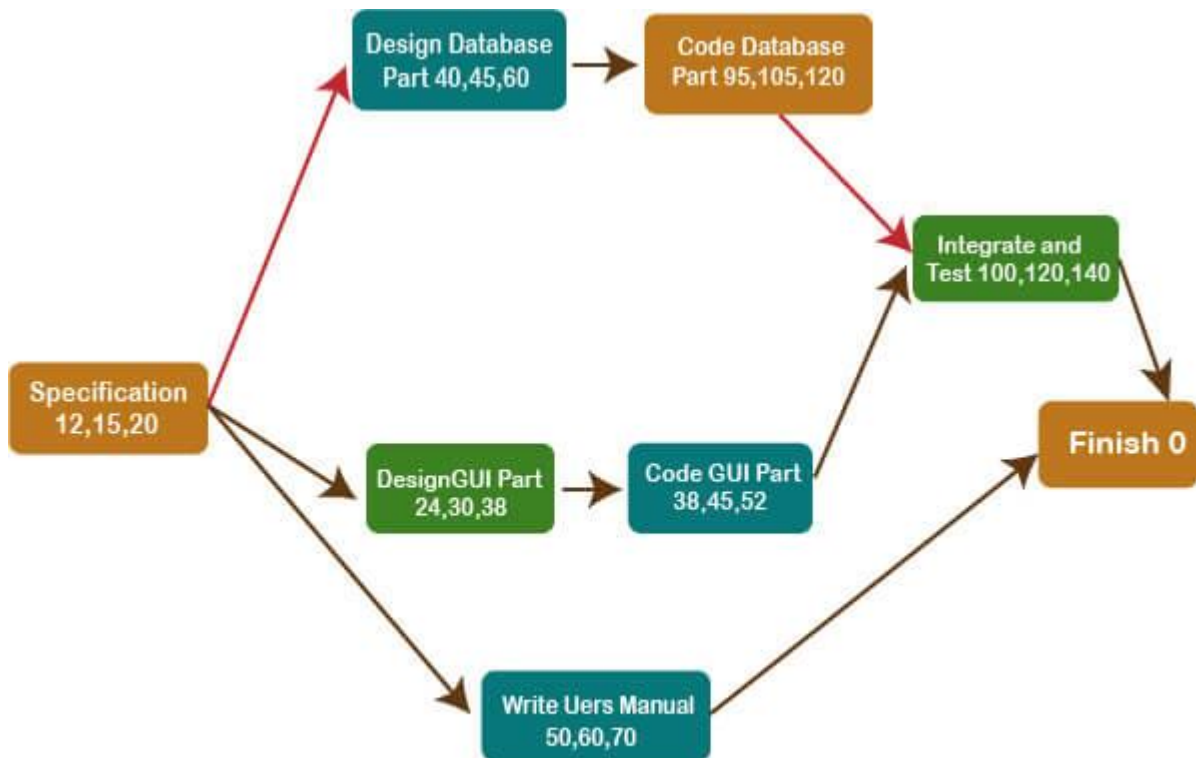
PERT chart representation of the MIS problem

Advantage of PERT

It forces the manager to plan.

It shows the interrelationships among the tasks in the project and, in particular, clearly identifies the critical path of the project, thus helping to focus on it.

It exposes all possible parallelism in the activities and thus helps in allocating resources.

It allows scheduling and stimulation of alternative schedules.

It enables the manager to monitor and control the project.

Logic Network

The Logic Network shows the order of activities over time. It shows the sequence in which activities are to do. Distinguishing events and pinning down the project are the two primary uses. Moreover, it will help with understanding task dependencies, a timescale, and overall project workflow.

Product Breakdown Structure

Product Breakdown Structure (BBS) is a management tool and necessary a part of the project designing. It's a task-oriented system for subdividing a project into product parts. The product breakdown structure describes subtasks or work packages and represents the connection

between work packages. Within the product breakdown Structure, the project work has diagrammatically pictured with various types of lists. The product breakdown structure is just like the work breakdown structure (WBS).

Work Breakdown Structure

Work Breakdown Structure (WBS) is used to decompose a given function set recursively into small activities. WBS provides a notation for representing the significant tasks that need to be carried out to solve a problem. The problem name labels the root of the tree. Each node of the tree is destroyed down into smaller activities that are building the children of the node. Each activity is recursively decomposed into smaller sub-activities until at the leaf level; the activities require approximately two weeks to develop, as shown in fig represents the WBS of MIS (Management Information System) software. While breaking down a function into smaller tasks, the manager has to make some hard decisions. If a task is broken down into a large number of minimal activities, these can be carried out independently. Thus, it becomes feasible to develop the product faster (with the help of additional workforce). Therefore, to be able to do a project in the least amount of time, the manager needs to break the vital function into smaller ones, expecting to find more parallelism. However, it is not useful to subdivide tasks into units which take less than a week or two to execute. Excellent subdivision measure that a disproportionate amount of time must be spent on preparing and revising various charts.

Resource Histogram

The resource histogram is precisely a bar chart that used for displaying the amounts of time that a resource is scheduled to be worked on over a prearranged and specific period. Resource histograms can also contain the related feature of resource availability, used for comparison on purposes of contrast.

Critical Path Analysis

Critical path analysis is a technique that is used to categorize the activities which are required to complete a task, as well as classifying the time which is needed to finish each activity and the relationships between the activities. It is also called a critical path method. CPA helps in predicting whether a project will expire on time. From the activity network representation following method can be made. The minimum time (MT) to complete the project is the maximum of all way from start to finish. The earliest start (ES) time of a method is the maximum of all paths from the start to the task. The current start time is the difference between MT and the maximum of all paths from this method to the finish. The earliest finish time (EF) of a function is the sum of the earliest start time of the function and the duration of the function. The latest finish (LF) time of a function can be obtained by subtracting maximum of all paths from this method to complete from MT. The slack time (ST) is LS - EF and equally can be indicated as LF - EF. The slack time (or float time) is the total time that a function may be delayed before it will affect the last time of the project. The slack time means the "flexibility" in the starting and completion of tasks. A critical method is one with a zero slack time. A path from the begin node to the last node containing only critical tasks is called a critical path. These parameters for various methods for the MIS problem are shown in the following table.

Task	ES	EF	LS	LF	ST
Specification	0	15	0	15	0
Design database	15	60	15	60	0
Design GUI part	15	45	90	120	75
Code database	60	165	60	165	0
Code GUI part	45	90	120	165	75
Integrate and test	165	285	165	285	0
Write user manual	15	75	225	285	210

Software Cost Estimation

For any new software project, it is necessary to know how much it will cost to develop and how much development time will it take. These estimates are needed before development is initiated, but how is this done? Several estimation procedures have been developed and are having the following attributes in common.

1. Project scope must be established in advanced.
2. Software metrics are used as a support from which evaluation is made.
3. The project is broken into small PCs which are estimated individually. To achieve true cost & schedule estimate, several option arise.
4. Delay estimation
5. Used symbol decomposition techniques to generate project cost and schedule estimates.
6. Acquire one or more automated estimation tools.

Uses of Cost Estimation

1. During the planning stage, one needs to choose how many engineers are required for the project and to develop a schedule.
2. In monitoring the project's progress, one needs to access whether the project is progressing according to the procedure and takes corrective action, if necessary.

Cost Estimation Models

A model may be static or dynamic. In a static model, a single variable is taken as a key element for calculating cost and time. In a dynamic model, all variable are interdependent, and there is no basic variable.
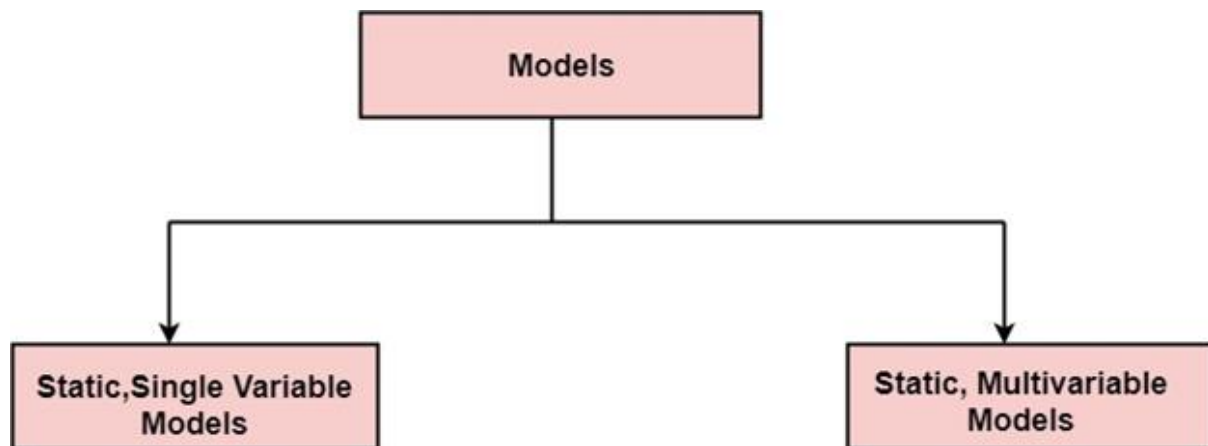
Static, Single Variable Models: When a model makes use of single variables to calculate desired values such as cost, time, efforts, etc. is said to be a single variable model. The most common equation is:

$C=aL^b$

Where   C = Costs
         L= size
         a and b are constants

The Software Engineering Laboratory established a model called SEL model, for estimating its software production. This model is an example of the static, single variable model.

        $E=1.4L^{0.93}$
        $DOC=30.4L^{0.90}$
        $D=4.6L^{0.26}$

Where   E= Efforts (Person Per Month)
         DOC=Documentation (Number of Pages)
         D = Duration (D, in months)
         L = Number of Lines per code

Static, Multivariable Models: These models are based on method (1), they depend on several variables describing various aspects of the software development environment. In some model, several variables are needed to describe the software development process, and selected equation combined these variables to give the estimate of time & cost. These models are called multivariable models.

WALSTON and FELIX develop the models at IBM provide the following equation gives a relationship between lines of source code and effort:

        $E=5.2L^{0.91}$

In the same manner duration of development is given by

D=4.1L$^{0.36}$

The productivity index uses 29 variables which are found to be highly correlated productivity as follows:

$$I = \sum_{i=1}^{29} W_i X_i$$

Where W_i is the weight factor for the i^{th}variable and $X_i=\{-1,0,+1\}$ the estimator gives $X_i$one of the values **-1, 0 or +1** depending on the variable decreases, has no effect or increases the productivity.

Example: Compare the Walston-Felix Model with the SEL model on a software development expected to involve 8 person-years of effort.

a.   Calculate the number of lines of source code that can be produced.
  b. Calculate the duration of the development.
  c. Calculate the productivity in LOC/PY
  d. Calculate the average manning

Solution:

The amount of manpower involved = 8PY=96persons-months

(a)Number of lines of source code can be obtained by reversing equation to give:

$$L = \left(\frac{E}{a}\right) 1/b$$

Then

L (SEL) = (96/1.4)1⁄0.93=94264 LOC
L (SEL) = (96/5.2)1⁄0.91=24632 LOC

(b)Duration in months can be calculated by means of equation

D (SEL) = 4.6 (L) 0.26
          = 4.6 (94.264)0.26 = 15 months
D (W-F) = 4.1 L$^{0.36}$
          = 4.1 (24.632)0.36 = 13 months

(c) Productivity is the lines of code produced per persons/month (year)

$$P\ (SEL) = \frac{94264}{8} = 11783\ \frac{LOC}{Person} - Years$$

$$P\ (Years) = \frac{24632}{8} = 3079\ \frac{LOC}{Person} - Years$$

(d) Average manning is the average number of persons required per month in the project

$$M\ (SEL) = \frac{96P-M}{15M} = 6.4\ Persons$$

$$M\ (W\text{-}F) = \frac{96P-M}{13M} = 7.4\ Persons$$

COCOMO Model

Boehm proposed COCOMO (Constructive Cost Estimation Model) in 1981.COCOMO is one of the most generally used software estimation models in the world. COCOMO predicts the efforts and schedule of a software product based on the size of the software.

The necessary steps in this model are:

1. Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code (KDLOC).
2. Determine a set of 15 multiplying factors from various attributes of the project.
3. Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step2.

The initial estimate (also called nominal estimate) is determined by an equation of the form used in the static single variable models, using KDLOC as the measure of the size. To determine the initial effort E_i in person-months the equation used is of the type is shown below

$E_i = a*(KDLOC)b$

The value of the constant a and b are depends on the project type.

In COCOMO, projects are categorized into three types:

1. Organic
2. Semidetached
3. Embedded

1.Organic: A development project can be treated of the organic type, if the project deals with developing a well-understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar methods of projects. **Examples of this type of projects are simple business systems, simple inventory management systems, and data processing systems.**

2. Semidetached: A development project can be treated with semidetached type if the development consists of a mixture of experienced and inexperienced staff. Team members may have finite experience in related systems but may be unfamiliar with some aspects of the order being developed. **Example of Semidetached system includes developing a new operating system (OS), a Database Management System (DBMS), and complex inventory management system.**

3. Embedded: A development project is treated to be of an embedded type, if the software being developed is strongly coupled to complex hardware, or if the stringent regulations on the operational method exist. **For Example:** ATM, Air Traffic control.

For three product categories, Bohem provides a different set of expression to predict effort (in a unit of person month)and development time from the size of estimation in KLOC(Kilo Line of code) efforts estimation takes into account the productivity loss due to holidays, weekly off, coffee breaks, etc.

According to Boehm, software cost estimation should be done through three stages:

1. Basic Model
2. Intermediate Model
3. Detailed Model

1. Basic COCOMO Model: The basic COCOMO model provide an accurate size of the project parameters. The following expressions give the basic COCOMO estimation model:

$$\text{Effort} = a_1 * (\text{KLOC}) \ a_2 \ \text{PM}$$
$$\text{Tdev} = b_1 * (\text{efforts}) b_2 \ \text{Months}$$

Where

KLOC is the estimated size of the software product indicate in Kilo Lines of Code,

a_1, a_2, b_1, b_2 are constants for each group of software products,

Tdev is the estimated time to develop the software, expressed in months,

Effort is the total effort required to develop the software product, expressed in **person months (PMs)**.

Estimation of development effort

For the three classes of software products, the formulas for estimating the effort based on the code size are shown below:

Organic: Effort = 2.4(KLOC) 1.05 PM

Semi-detached: Effort = 3.0(KLOC) 1.12 PM

Embedded: Effort = 3.6(KLOC) 1.20 PM
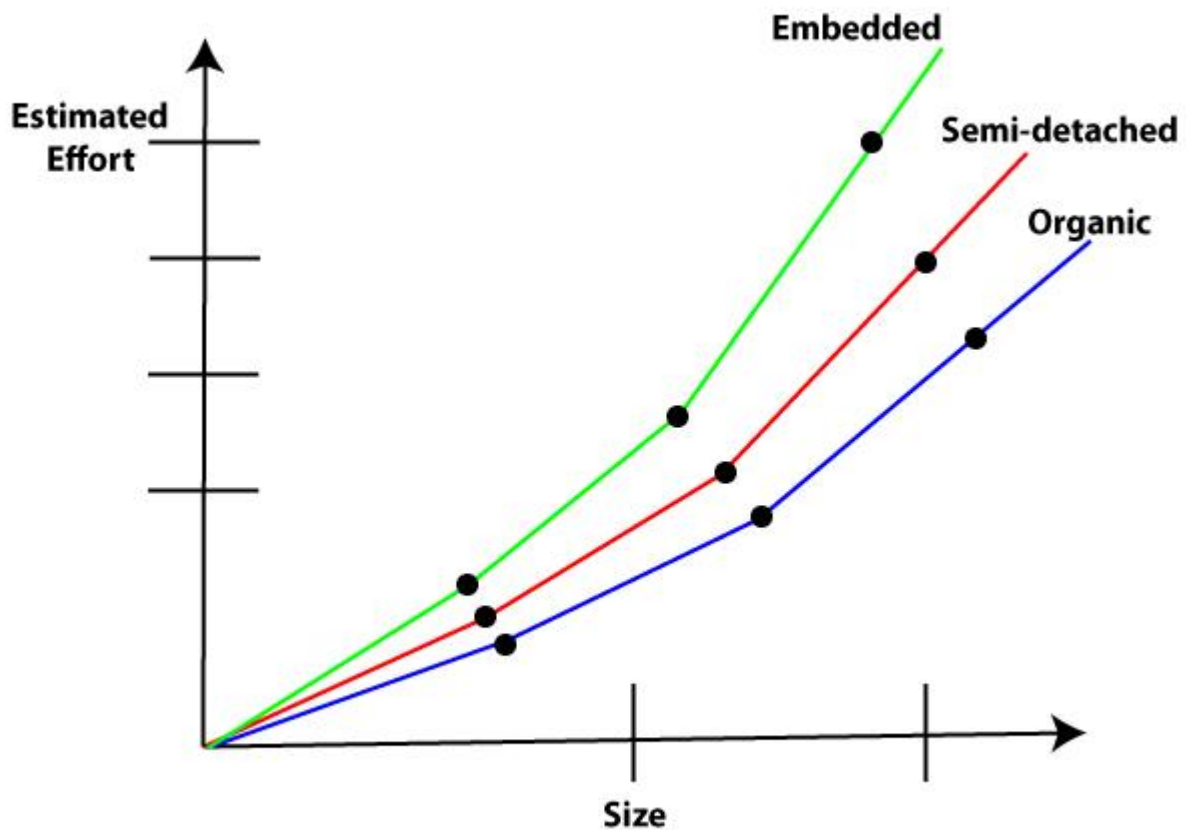
Estimation of development time

For the three classes of software products, the formulas for estimating the development time based on the effort are given below:

Organic: Tdev = 2.5(Effort) 0.38 Months

Semi-detached: Tdev = 2.5(Effort) 0.35 Months

Embedded: Tdev = 2.5(Effort) 0.32 Months

Some insight into the basic COCOMO model can be obtained by plotting the estimated characteristics for different software sizes. Fig shows a plot of estimated effort versus product size. From fig, we can observe that the effort is somewhat superliner in the size of the software product. Thus, the effort required to develop a product increases very rapidly with project size.

Effort versus product size

The development time versus the product size in KLOC is plotted in fig. From fig it can be observed that the development time is a sub linear function of the size of the product, i.e. when the size of the product increases by two times, the time to develop the product does not double but rises moderately. This can be explained by the fact that for larger products, a larger number of activities which can be carried out concurrently can be identified. The parallel activities can be carried out simultaneously by the engineers. This reduces the time to complete the project.

Further, from fig, it can be observed that the development time is roughly the same for all three categories of products. For example, a 60 KLOC program can be developed in approximately 18 months, regardless of whether it is of organic, semidetached, or embedded type.
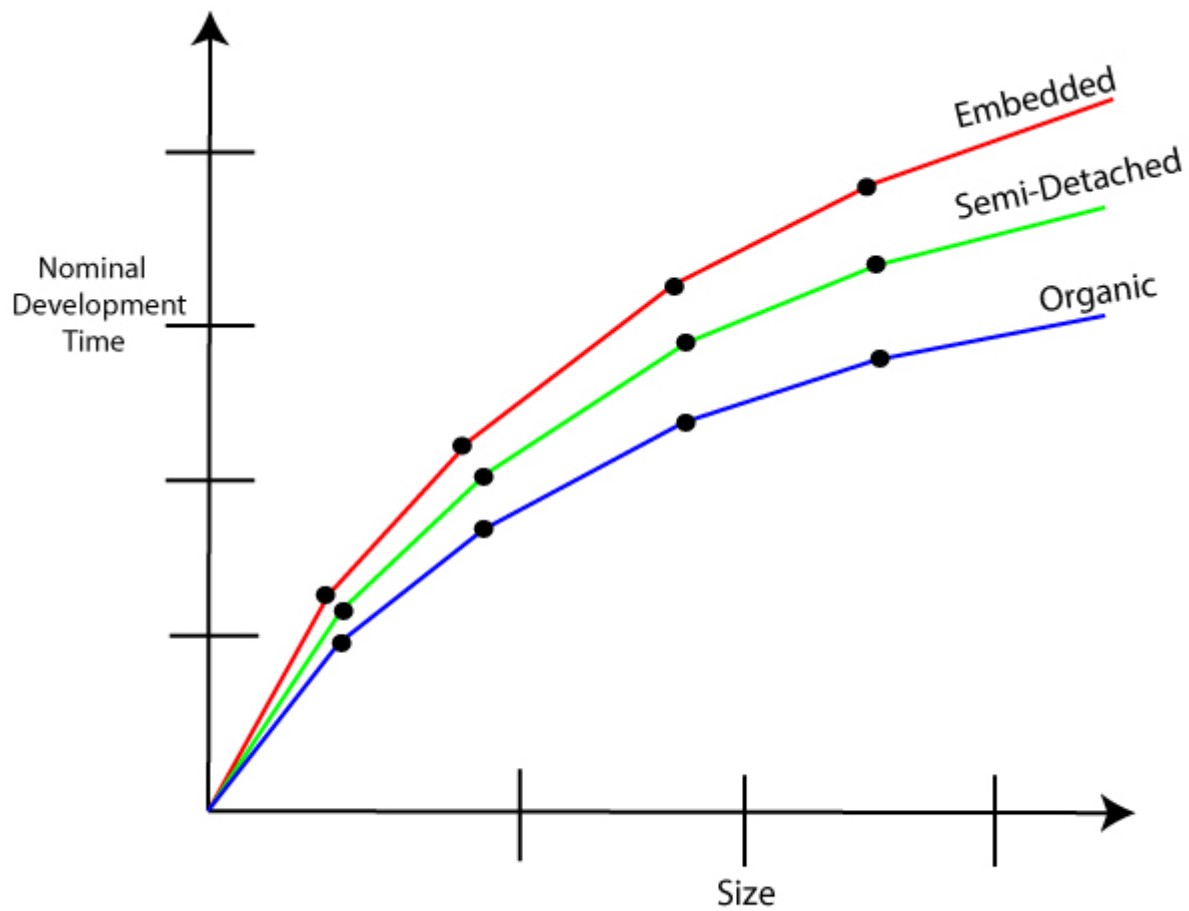
Development time versus size

From the effort estimation, the project cost can be obtained by multiplying the required effort by the manpower cost per month. But, implicit in this project cost computation is the assumption that the entire project cost is incurred on account of the manpower cost alone. In addition to manpower cost, a project would incur costs due to hardware and software required for the project and the company overheads for administration, office space, etc.

It is important to note that the effort and the duration estimations obtained using the COCOMO model are called a nominal effort estimate and nominal duration estimate. The term nominal implies that if anyone tries to complete the project in a time shorter than the estimated duration, then the cost will increase drastically. But, if anyone completes the project over a longer period of time than the estimated, then there is almost no decrease in the estimated cost value.

Example1: Suppose a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three model i.e., organic, semi-detached & embedded.

Solution: The basic COCOMO equation takes the form:

$$Effort=a_1*(KLOC) \; a_2 \; PM$$
$$Tdev=b_1*(efforts)b_2 \; Months$$
Estimated Size of project= 400 KLOC

(i)Organic Mode

$$E = 2.4 * (400)1.05 = 1295.31 \; PM$$
$$D = 2.5 * (1295.31)0.38=38.07 \; PM$$

(ii)Semidetached Mode

$$E = 3.0 * (400)1.12=2462.79 \; PM$$
$$D = 2.5 * (2462.79)0.35=38.45 \; PM$$

(iii) Embedded Mode

$$E = 3.6 * (400)1.20 = 4772.81 \; PM$$
$$D = 2.5 * (4772.8)0.32 = 38 \; PM$$

Example2: A project size of 200 KLOC is to be developed. Software development team has average experience on similar type of projects. The project schedule is not very tight. Calculate the Effort, development time, average staff size, and productivity of the project.

Solution: The semidetached mode is the most appropriate mode, keeping in view the size, schedule and experience of development time.

Hence $E=3.0(200)1.12=1133.12PM$
$D=2.5(1133.12)0.35=29.3PM$

$$Average \; Staff \; Size \; (SS) = \frac{E}{D} \; Persons$$

$$= \frac{1133.12}{29.3} = 38.67 \; Persons$$

$$Productivity = \frac{KLOC}{E} = \frac{200}{1133.12} = 0.1765 \; KLOC/PM$$

$$P = 176 \; LOC/PM$$

2. Intermediate Model: The basic Cocomo model considers that the effort is only a function of the number of lines of code and some constants calculated according to the various software systems. The intermediate COCOMO model recognizes these facts and refines the initial estimates obtained through the basic COCOMO model by using a set of 15 cost drivers based on various attributes of software engineering.

Classification of Cost Drivers and their attributes:

(i) Product attributes -

- o   Required software reliability extent

- Size of the application database
- The complexity of the product

Hardware attributes -

- Run-time performance constraints
- Memory constraints
- The 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

The cost drivers are divided into four categories:

Cost Drivers	RATINGS					
	Very low	Low	Nominal	High	Very High	Extra High
Product Attributes						
RELY	0.75	0.88	1.00	1.15	1.40	..
DATA	..	0.94	1.00	1.08	1.16	..
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
Computer Attributes						
TIME	..	..	1.00	1.11	1.30	1.66
STOR	..	..	1.00	1.06	1.21	1.56
VIRT	..	0.87	1.00	1.15	1.30	..
TURN	..	0.87	1.00	1.07	1.15	..

Cost Drivers	RATINGS					
	Very low	Low	Nominal	High	Very high	Extra high
Personnel Attributes						
ACAP	1.46	1.19	1.00	0.86	0.71	..
AEXP	1.29	1.13	1.00	0.91	0.82	..
PCAP	1.42	1.17	1.00	0.86	0.70	..
VEXP	1.21	1.10	1.00	0.90	..	..
LEXP	1.14	1.07	1.00	0.95	..	..
Project Attributes						
MODP	1.24	1.10	1.00	0.91	0.82	..
TOOL	1.24	1.10	1.00	0.91	0.83	..
SCED	1.23	1.08	1.00	1.04	1.10	..

Intermediate COCOMO equation:

$$E = a_i (KLOC) \, b_i * EAF$$
$$D = c_i (E) d_i$$

Coefficients for intermediate COCOMO

Project	a_i	b_i	c_i	d_i
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

3. Detailed COCOMO Model: Detailed COCOMO incorporates all qualities of the standard version with an assessment of the cost driver?s effect on each method of the software engineering process. The detailed model uses various effort multipliers for each cost driver

property. In detailed cocomo, the whole software is differentiated into multiple modules, and then we apply COCOMO in various modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:

1. Planning and requirements
2. System structure
3. Complete structure
4. Module code and test
5. Integration and test
6. Cost Constructive model

The effort is determined as a function of program estimate, and a set of cost drivers are given according to every phase of the software lifecycle.

Software Project Planning

A Software Project is the complete methodology of programming advancement from requirement gathering to testing and support, completed by the execution procedures, in a specified period to achieve intended software product.

Need of Software Project Management

Software development is a sort of all new streams in world business, and there's next to no involvement in structure programming items. Most programming items are customized to accommodate customer's necessities. The most significant is that the underlying technology changes and advances so generally and rapidly that experience of one element may not be connected to the other one. All such business and ecological imperatives bring risk in software development; hence, it is fundamental to manage software projects efficiently.

Software Project Manager

Software manager is responsible for planning and scheduling project development. They manage the work to ensure that it is completed to the required standard. They monitor the progress to check that the event is on time and within budget. The project planning must incorporate the major issues like size & cost estimation scheduling, project monitoring, personnel selection evaluation & risk management. To plan a successful software project, we must understand:

o Scope of work to be completed
o Risk analysis
o The resources mandatory
o The project to be accomplished
o Record of being followed

Software Project planning starts before technical work start. The various steps of planning activities are:
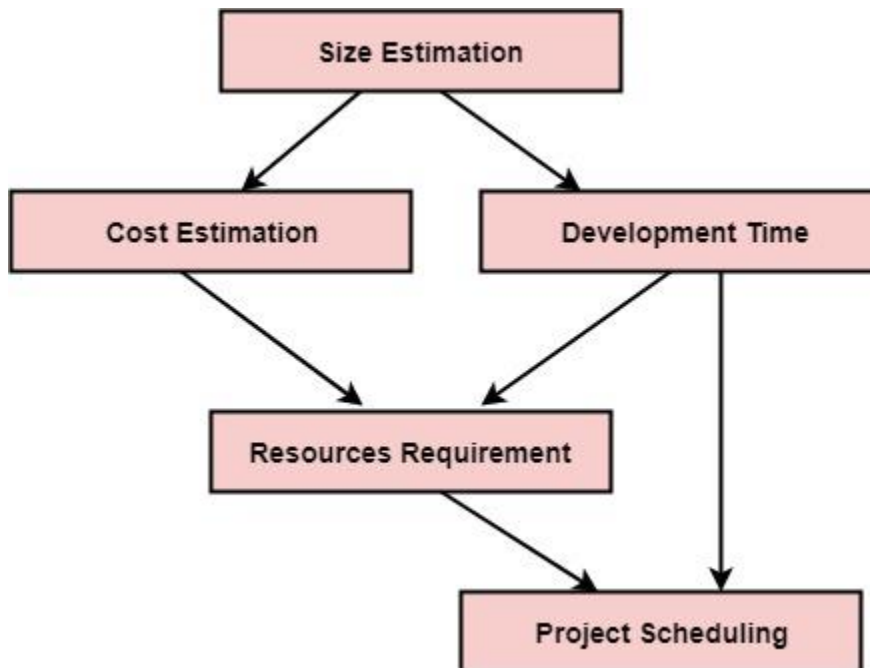
The size is the crucial parameter for the estimation of other activities. Resources requirement are required based on cost and development time. Project schedule may prove to be very useful for controlling and monitoring the progress of the project. This is dependent on resources & development time.