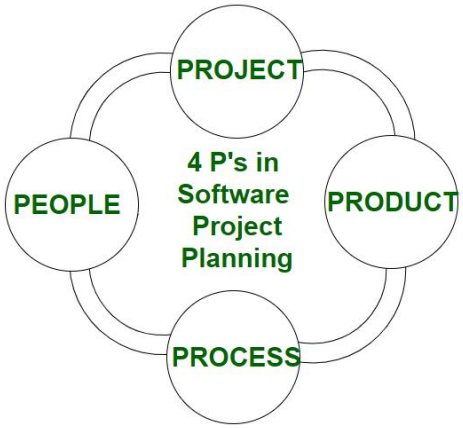
UNIT 5

For properly building a product, there's a very important concept that we all should know in software project planning while developing a product. There are 4 critical components in software project planning which are known as the **4P's** namely:

- Product
- Process
- People
- Project

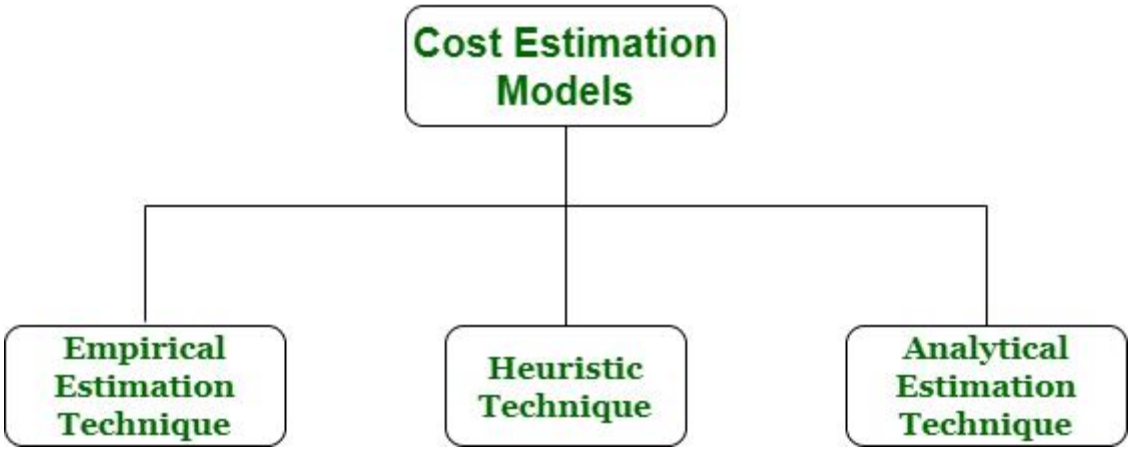


These components play a very important role in your project that can help your team meet its goals and objective. Now, Let's dive into each of them a little in detail to get a better understanding:

- **People**
  The most important component of a product and its successful implementation is human resources. In building a proper product, a well-managed team with clear-cut roles defined for each person/team will lead to the success of the product. We need to have a good team in order to save our time, cost, and effort. Some assigned roles in software project planning are **project manager, team leaders, stakeholders, analysts,** and other **IT professionals**. Managing people successfully is a tricky process which a good project manager can do.
- **Product**
  As the name inferred, this is the deliverable or the result of the project. The project manager should clearly define the product scope to ensure a successful result, control the team members, as well technical hurdles that he or she may encounter during the building of a product. The product can consist of both tangible or intangible such as shifting the company to a new place or getting a new software in a company.
- **Process**
  In every planning, a clearly defined process is the key to the success of any product. It regulates how the team will go about its development in the respective time period. The Process has several steps involved like, documentation phase, implementation phase, deployment phase, and interaction phase.

- **Project**
  The last and final P in software project planning is Project. It can also be considered as a blueprint of process. In this phase, the project manager plays a critical role. They are responsible to guide the team members to achieve the project's target and objectives, helping & assisting them with issues, checking on cost and budget, and making sure that the project stays on track with the given deadlines.

Cost estimation simply means a technique that is used to find out the cost estimates. The cost estimate is the financial spend that is done on the efforts to develop and test software in Software Engineering. Cost estimation models are some mathematical algorithms or parametric equations that are used to estimate the cost of a product or a project.
Various techniques or models are available for cost estimation, also known as Cost Estimation Models as shown below :



1. **Empirical Estimation Technique –**
   Empirical estimation is a technique or model in which empirically derived formulas are used for predicting the data that are a required and essential part of the software project planning step. These techniques are usually based on the data that is collected previously from a project and also based on some guesses, prior experience with the development of similar types of projects, and assumptions. It uses the size of the software to estimate the effort.
   In this technique, an educated guess of project parameters is made. Hence, these models are based on common sense. However, as there are many activities involved in empirical estimation techniques, this technique is formalized. For example Delphi technique and Expert Judgement technique.

2. **Heuristic Technique –**
   Heuristic word is derived from a Greek word that means "to discover". The heuristic technique is a technique or model that is used for solving problems, learning, or discovery in the practical methods which are used for achieving immediate goals. These techniques are flexible and simple for taking quick decisions through shortcuts and good enough calculations, most probably when working with complex data. But the decisions that are made using this technique are necessary to be optimal.

In this technique, the relationship among different project parameters is expressed using mathematical equations. The popular heuristic technique is given by Constructive Cost Model (COCOMO). This technique is also used to increase or speed up the analysis and investment decisions.

3. **Analytical Estimation Technique –**
   Analytical estimation is a type of technique that is used to measure work. In this technique, firstly the task is divided or broken down into its basic component operations or elements for analyzing. Second, if the standard time is available from some other source, then these sources are applied to each element or component of work.
   Third, if there is no such time available, then the work is estimated based on the experience of the work. In this technique, results are derived by making certain basic assumptions about the project. Hence, the analytical estimation technique has some scientific basis. Halstead's software science is based on an analytical estimation model.

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

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

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

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

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

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.

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

What is Software Configuration Management?

In Software Engineering, **Software Configuration Management(SCM)** is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. The primary goal is to increase productivity with minimal mistakes. SCM is part of cross-disciplinary field of configuration management and it can accurately determine who made which revision.

Software Quality Assurance

What is Quality?

Quality defines to any measurable characteristics such as correctness, maintainability, portability, testability, usability, reliability, efficiency, integrity, reusability, and interoperability.

There are two kinds of Quality:

Quality of Design: Quality of Design refers to the characteristics that designers specify for an item. The grade of materials, tolerances, and performance specifications that all contribute to the quality of design.

Quality of conformance: Quality of conformance is the degree to which the design specifications are followed during manufacturing. Greater the degree of conformance, the higher is the level of quality of conformance.

Software Quality: Software Quality is defined as the conformance to explicitly state functional and performance requirements, explicitly documented development standards, and inherent characteristics that are expected of all professionally developed software.

Quality Control: Quality Control involves a series of inspections, reviews, and tests used throughout the software process to ensure each work product meets the requirements place upon it. Quality control includes a feedback loop to the process that created the work product.

Quality Assurance: Quality Assurance is the preventive set of activities that provide greater confidence that the project will be completed successfully.

Quality Assurance focuses on how the engineering and management activity will be done?

As anyone is interested in the quality of the final product, it should be assured that we are building the right product.

It can be assured only when we do inspection & review of intermediate products, if there are any bugs, then it is debugged. This quality can be enhanced.

Importance of Quality

We would expect the quality to be a concern of all producers of goods and services. However, the distinctive characteristics of software and in particular its intangibility and complexity, make special demands.

Increasing criticality of software: The final customer or user is naturally concerned about the general quality of software, especially its reliability. This is increasing in the case as organizations become more dependent on their computer systems and software is used more and more in safety-critical areas. For example, to control aircraft.

The intangibility of software: This makes it challenging to know that a particular task in a project has been completed satisfactorily. The results of these tasks can be made tangible by demanding that the developers produce 'deliverables' that can be examined for quality.

Accumulating errors during software development: As computer system development is made up of several steps where the output from one level is input to the next, the errors in the earlier deliverables will be added to those in the later stages leading to accumulated determinable effects. In general the later in a project that an error is found, the more expensive it will be to fix. In addition, because the number of errors in the system is unknown, the debugging phases of a project are particularly challenging to control.

Software Quality Assurance

Software quality assurance is a planned and systematic plan of all actions necessary to provide adequate confidence that an item or product conforms to establish technical requirements.

A set of activities designed to calculate the process by which the products are developed or manufactured.

SQA Encompasses

- A quality management approach
- Effective Software engineering technology (methods and tools)
- Formal technical reviews that are tested throughout the software process
- A multitier testing strategy
- Control of software documentation and the changes made to it.
- A procedure to ensure compliances with software development standards
- Measuring and reporting mechanisms.

SQA Activities

Software quality assurance is composed of a variety of functions associated with two different constituencies ? the software engineers who do technical work and an SQA group that has responsibility for quality assurance planning, record keeping, analysis, and reporting.

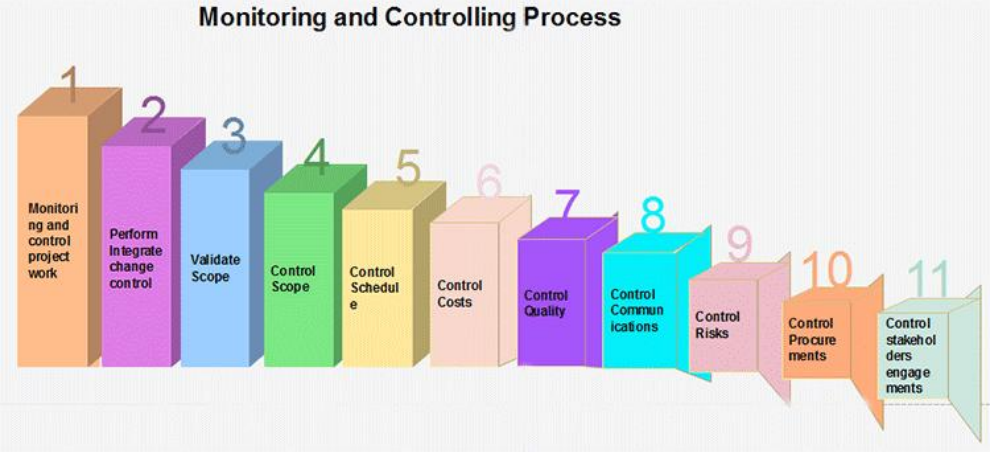
Following activities are performed by an independent SQA group:

1. **Prepares an SQA plan for a project:** The program is developed during project planning and is reviewed by all stakeholders. The plan governs quality assurance activities performed by the software engineering team and the SQA group. The plan identifies calculation to be performed, audits and reviews to be performed, standards that apply to the project, techniques for error reporting and tracking, documents to be produced by the SQA team, and amount of feedback provided to the software project team.
2. **Participates in the development of the project's software process description:** The software team selects a process for the work to be performed. The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g. ISO-9001), and other parts of the software project plan.
3. **Reviews software engineering activities to verify compliance with the defined software process:** The SQA group identifies, reports, and tracks deviations from the process and verifies that corrections have been made.
4. **Audits designated software work products to verify compliance with those defined as a part of the software process:** The SQA group reviews selected work products, identifies, documents and tracks deviations, verify that corrections have been made, and periodically reports the results of its work to the project manager.
5. **Ensures that deviations in software work and work products are documented and handled according to a documented procedure:** Deviations may be encountered in the project method, process description, applicable standards, or technical work products.
6. **Records any noncompliance and reports to senior management:** Non- compliance items are tracked until they are resolved.

Project Monitoring and Control

Monitoring and Controlling are processes needed to track, review, and regulate the progress and performance of the project. It also identifies any areas where changes to the project management method are required and initiates the required changes.

The Monitoring & Controlling process group includes eleven processes, which are:



1. **Monitor and control project work:** The generic step under which all other monitoring and controlling activities fall under.
2. **Perform integrated change control:** The functions involved in making changes to the project plan. When changes to the schedule, cost, or any other area of the project management plan are necessary, the program is changed and re-approved by the project sponsor.
3. **Validate scope:** The activities involved with gaining approval of the project's deliverables.
4. **Control scope:** Ensuring that the scope of the project does not change and that unauthorized activities are not performed as part of the plan (scope creep).
5. **Control schedule:** The functions involved with ensuring the project work is performed according to the schedule, and that project deadlines are met.
6. **Control costs:** The tasks involved with ensuring the project costs stay within the approved budget.
7. **Control quality:** Ensuring that the quality of the project?s deliverables is to the standard defined in the project management plan.
8. **Control communications:** Providing for the communication needs of each project stakeholder.
9. **Control Risks:** Safeguarding the project from unexpected events that negatively impact the project's budget, schedule, stakeholder needs, or any other project success criteria.
10. **Control procurements:** Ensuring the project's subcontractors and vendors meet the project goals.
11. **Control stakeholder engagement:** The tasks involved with ensuring that all of the project's stakeholders are left satisfied with the project work.

Management Role in Software Development

The project manager has an important role to play in any software development project. Let’s explore.

- Develops, manages, and prepares the best development team.

- Guides, coaches, and mentors software developers and engineers.
- Provide technical leadership and project management for every aspect of the software.
- Supervises the architecture, as well as lead efforts to build a technical roadmap for all projects.
- Prepare lifecycle for various projects, which include doing research, designing, developing, evaluating, and testing, together with product management delivery.
- Establish and stimulate standards and processes in software development, together with the best practices to deliver high quality and scalable software.
- Build relations with prospective and existing internal customers to interpret all of the individual needs and requirements.
- Closely work with developers, engineers, and product management throughout the company to influence the development of the solution to boost products.
- Ensuring top quality design reviews that attain business goals.
- Supervise allocation of resources to make sure that the business and personnel development goals are attained.
- Involved in strategic planning to accomplish technical and leadership chain with customers.
- Learn how products add value to the respective businesses.
- Evaluating projects, developing and updating schedules, and supervising the status of the project.
- Manages and executes the development projects from beginning to end.
- Effective collaboration with all the team members, and holding regular team meetings.

The Significance of Management in Software Development

In any software development company, the role of management in software development is becoming more significant. IT companies of all shapes and sizes are on the lookout to widen their scope of operations to acquire high competence in the highly competitive market. Thus, project managers are high in demand in the industry to help companies build software solutions and execute numerous projects for various customers in a timely and effective manner.

Single Point of Contact between Developers and Customers

Maintaining effective communication between developers and customers is paramount. One could not expect developers to articulate the status of the work or any details that could come up halfway through the process of building. The manager articulates the requirements of the client to the developers and vice versa, thus he/she serves as the central link between developers and customers by coordinating and transferring information back and forth.

Furthermore, management schedules meetings, manage customer relationships, reiterates the requirements of a project to developers as well as the clients.

Establishing a Clear Project Management Scope

Outlining a clear project scope clarifies the goal of the end-user of the software solution that helps form the process accordingly. Aside from that, the scope is a measure of the requirements as well,

which should be delivered to the customer successfully. Determining the project scope ensures that everything required to complete a project successfully is in order.

Project managers give due emphasis to project management scope. It's formulated and crated beforehand, during the planning stage, and serves as the foundation of all the following stages. The costs, resources, and project delivery time are paramount.

The scope is completely dependent on them, meaning that any changes in fact could also make influences. Nonetheless, the role of the project manager is to leave unaltered the scope from all the changing factors and proceed with the development.

Assures Extensive Quality Control over the Process and Product
Another defining aspect of project management is sticking to quality. It means delivering a software product with excellent levels of quality. During development, managers work hand-in-hand with developers, the design team, QA specialists, and others to ensure that the right quality standards are maintained by the team in the process and the finished product.

Education and Skillsets
In general, software development management requires a bachelor's degree in computer engineering, computer science, or other related fields. Some employers, however, may require a Master of Science degree or an MBA for senior-level positions. Furthermore, a lot of employers also require comprehensive experience in software designing and development, working knowledge on the different programming languages and platforms, like Java for instance. Another great plus is having experience in leading development teams and projects.

Software development management in specialized industries, like finance or medical research may also need education and experience in a particular industry.

Software development managers should have robust analytical and technical skills, with expertise in software platforms, languages, and the current methodologies. Moreover, they should have strong leadership, budgeting, and managerial abilities, which could include the ability of hiring, training, and evaluating the staff via performance reviews. Excellent verbal and communication skills are necessary for collaboration with the different management levels, determine the software requirements, and deliver effective solutions.

Computer aided software engineering (CASE) is the implementation of computer facilitated tools and methods in software development. CASE is used to ensure a high-quality and defect-free software. CASE ensures a check-pointed and disciplined approach and helps designers, developers, testers, managers and others to see the project milestones during development.
CASE can also help as a warehouse for documents related to projects, like business plans, requirements and design specifications. One of the major advantages of using CASE is the delivery of the final product, which is more likely to meet real-world requirements as it ensures that customers remain part of the process.

CASE illustrates a wide set of labor-saving tools that are used in software development. It generates a framework for organizing projects and to be helpful in enhancing productivity. There was more interest in the concept of CASE tools years ago, but less so today, as the tools have

morphed into different functions, often in reaction to software developer needs. The concept of CASE also received a heavy dose of criticism after its release.

CASE Tools:
The essential idea of CASE tools is that in-built programs can help to analyze developing systems in order to enhance quality and provide better outcomes. Throughout the 1990, CASE tool became part of the software lexicon, and big companies like IBM were using these kinds of tools to help create software.
Various tools are incorporated in CASE and are called CASE tools, which are used to support different stages and milestones in a software development life cycle.

Types of CASE Tools:

1. **Diagramming Tools:**
   It helps in diagrammatic and graphical representations of the data and system processes. It represents system elements, control flow and data flow among different software components and system structure in a pictorial form.
   For example, Flow Chart Maker tool for making state-of-the-art flowcharts.

2. **Computer Display and Report Generators:**
   It helps in understanding the data requirements and the relationships involved.

3. **Analysis Tools:**
   It focuses on inconsistent, incorrect specifications involved in the diagram and data flow. It helps in collecting requirements, automatically check for any irregularity, imprecision in the diagrams, data redundancies or erroneous omissions.
   For example,
   - (i) Accept 360, Accompa, CaseComplete for requirement analysis.
   - (ii) Visible Analyst for total analysis.

4. **Central Repository:**
   It provides the single point of storage for data diagrams, reports and documents related to project management.

5. **Documentation Generators:**
   It helps in generating user and technical documentation as per standards. It creates documents for technical users and end users.
   For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.

6. **Code Generators:**
   It aids in the auto generation of code, including definitions, with the help of the designs, documents and diagrams.

Advantages of the CASE approach:

- As special emphasis is placed on redesign as well as testing, the servicing cost of a product over its expected lifetime is considerably reduced.

- The overall quality of the product is improved as an organized approach is undertaken during the process of development.

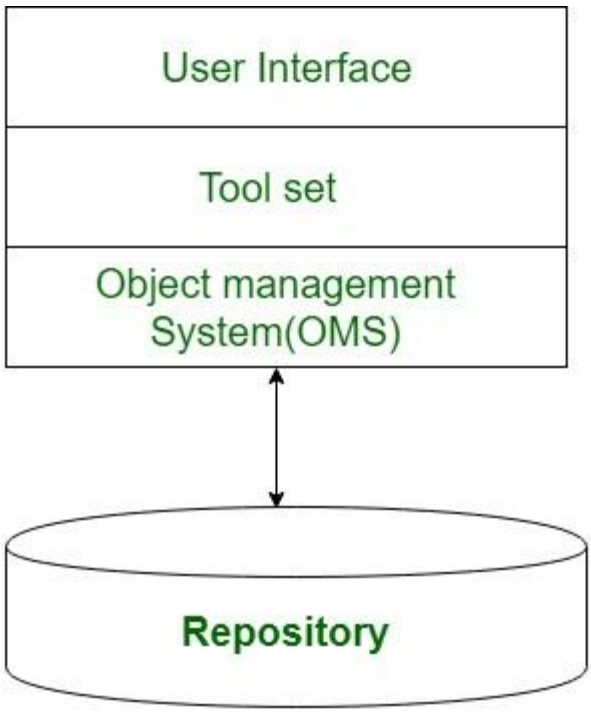
- Chances to meet real-world requirements are more likely and easier with a computer-aided software engineering approach.

- CASE indirectly provides an organization with a competitive advantage by helping ensure the development of high-quality products.

Disadvantages of the CASE approach:

- **Cost:** Using case tool is a very costly. Mostly firms engaged in software development on a small scale do not invest in CASE tools because they think that the benefit of CASE are justifiable only in the development of large systems.
- **Learning Curve:** In most cases, programmers productivity may fall in the initial phase of implementation , because user need time to learn the technology. Many consultants offer training and on-site services that can be important to accelerate the learning curve and to the development and use of the CASE tools.
- **Tool Mix:** It is important to build an appropriate selection tool mix to urge cost advantage CASE integration and data integration across all platforms is extremely important.

The design of a typical trendy CASE (Computer power-assisted software package Engineering) atmosphere is shown graphically below. The vital elements of a contemporary CASE atmosphere are a computer program, toolset, object management system (OMS), and a repository. The characteristics of a toolset are mentioned earlier.



Architecure of a modern CASE environment

User Interface:
the user interface provides a regular framework for accessing the various tools so creating it easier for the users to act with the different tools and reducing the overhead of learning however the different tools are used.

Object Management System (OMS) and Repository:
Different case tools represent the product as a group of entities like specification, design, text data, project arrange, etc. the thing management system maps these logical entities such into the underlying storage management system (repository).

The industrial on-line database management systems are meshed towards supporting giant volumes of data structured as straightforward comparatively short records. There are some forms of entities however sizable amount of instances. in contrast, CASE tools produce an oversized range of entity and relation varieties with maybe some instances of every. so the thing management system takes care of befittingly mapping into the underlying storage management system.