SOFTWARE ENGINEERING

Unit- 1

Software:- software is a collection of instructions that enable the user to interact with a computer, its hardware, or perform tasks.

Software life cycle: -

1. Planning
2. Defining
3. Designing
4. Building
5. Testing
6. Deployment

Software Engineering:-

- The **software** is a collection of integrated programs.
- **Software** subsists of carefully-organized instructions and code written by developers on any of various particular computer languages.
- Computer programs and related documentation such as requirements, design models and user manuals.
- **Engineering** is the application of scientific and practical knowledge to invent, design, build, maintain, and improve frameworks, processes, etc.
- **Software Engineering** is an engineering branch related to the evolution of software product using well-defined scientific principles, techniques, and procedures. The result of software engineering is an effective and reliable software product.

Characteristics: -

1. **Functionality: -** The functionality of software refers to its ability to perform and function according to design specification. In simple terms, software systems should function correctly, i.e. perform all the functions for which they are designed.

2. **Usability (User-friendly):-** The user-friendliness of the software is characterized by its ease of use. In other words, learning how to use the software should require less effort or time. Navigating the software is extremely important since it helps determine the journey the user takes within the software.
3. **Flexibility:-** Software Flexibility refers to the ability of the software solution to adapt to potential or future changes in its requirements. When evaluating the flexibility of software, look at how simple it is to add, modify, or remove features without interfering with the current operation.
4. **Efficiency:-** it refers to the software's ability to utilize human and system resources such as time, effort, CPU, memory, computation power, network bandwidth, files, databases, etc., as effectively and efficiently as possible.
5. **Reliability:-** The reliability of a software product describes that it will operate without failure over a specified period of time under certain conditions. It determines the ability of software to maintain its level of performance under specified conditions for a specified period of time.
6. **Maintainability:-** Maintainability refers to how easily you can repair, improve and comprehend software code. In some ways, maintaining is similar to being flexible. Maintainability deals with the modification of errors and minor alterations to software code, while flexibility focuses on major functional extensions. It also involves maintaining the services and functionality of the software.
7. **Portability:-** Portability refers to the ability to use software in different environments. This is the ease with which software can be ported from one platform to another without (or with minimal) changes, while obtaining similar results.
8. **Integrity:-** Integrity is key for demonstrating the safety, security, and maintainability of your software. Some people tend to associate integrity with security, believing it is resistant to hacks and privacy violations. To others, high integrity means that the software cannot be modified without authorization.

Components of Software :-

- A software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition.
- There are three components of the software: These are : Program, Documentation, and Operating Procedures.

1. **Program:-** A computer program is a list of instructions that tell a computer what to do.
2. **Documentation:-** Source information about the product contained in design documents, detailed code comments, etc.
3. **Operating Procedures:-** Set of step-by-step instructions compiled by an organization to help workers carry out complex routine operations.

There are four basic key process activities:

1. **Software Specifications :-** In this process, detailed description of a software system to be developed with its functional and non-functional requirements.
2. **Software Development :-** In this process, designing, programming, documenting, testing, and bug fixing is done.
3. **Software Validation :-** In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs.
4. **Software Evolution :-** It is a process of developing software initially, then timely updating it for various reasons.

Applications: -

1. Word Processing Software

Word processing software is used to format, beautify, and manipulate text. It allows features such as synonyms and antonyms. You can change the fonts, colors, and style according to your choice with the word art feature.Errorchecking as well as grammar and spell checking options are also available in it. Microsoft Word is the best example of a word processing software.

2. Spreadsheet Software

Spreadsheet software is majorly used to store data in table format and perform calculations. Intersecting cells are given in a spreadsheet to keep various data fields such as time, date, text, and numbers. Users can perform calculations with formulas and functions. The best example of spreadsheet software is Microsoft Excel.

3. Presentation Software

Presentation Software lets you put forth your thoughts and ideas in a piece of visual information. Then, you can present that information in the form of slides. You can make your slides interactive and informative by adding videos, texts, charts, graphs, and images. The best example of presentation software is Microsoft PowerPoint.

4. Multimedia Software

Multimedia Software lets you create or record videos, audio, and image files. Such app software is used in video editing, graphics, and animations. Common examples of multimedia software are VLC player, MX Player, and Windows Media Player.

5. Web Browsers

These software applications are used to browse on the internet. They let you locate and retrieve data from the web. The most popular web browsers are Chrome and Firefox.

6. Educational Software

These types of application software are called academic software as they are particularly designed to facilitate learning. All different kinds of tutorial software are included in it. Examples of educational software are EDX, MindPlay, and Kid Pix.

7. Graphics Software

Graphics Software is used to make changes in visual data, images, and animation. It comprises different editorial software. Adobe Photoshop, Unity 3d, and PaintShop are examples of graphics software.

8. Freeware

As you can guess from the name, this type of software is available free of cost. Therefore, you can download and install them for free. However, you are not allowed to make any change in its source code. Skype is an example of freeware software.

9. Shareware

Such softwares are distributed to the users on a trial basis. Then, if the users like it and want to continue, they have to pay for that software. An example of shareware software is Winzip.

10. Simulation Software

Simulation Software is a monitoring program that allows the user to observe an operation without performing it. Such software is useful when the existing system's work is not highly accurate, predictable, or dangerous. It is used widely in engineering, robotics, flight systems, weather forecast, testing, education, and video gaming. MATLAB is the best example of simulation software.

11. Open Source

Open Source software is available with a source code and rights for anyone to inspect, modify and enhance it. Moreover, most open-source software is available for free, while very few are paid ones at such a conditional level.

12. Closed Source

Closed Source software is precisely the opposite to open-source software. They are paid software and have intellectual property rights or patient over source code. It usually comes with some restrictions as well as terms and conditions

The Software Process Model:

A software process model is a specified definition of a software process, which is presented from a particular perspective. Models, by their nature, are a simplification, so a software process model is an abstraction of the actual process, which is being described. Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering.

1. Waterfall Model:-

Winston Royce introduced the Waterfall Model in 1970.This model has five phases: Requirements analysis and specification, design, implementation, and unit testing, integration and system testing, and operation and maintenance. The steps always follow in this order and do not overlap. The developer must complete every phase before the next phase begins. This model is named "Waterfall Model", because its diagrammatic representation resembles a cascade of waterfalls.

1. Requirements analysis and specification phase:

The aim of this phase is to understand the exact requirements of the customer and to document them properly. Both the customer and the software developer work together so as to document all the functions, performance, and interfacing requirement of the software. It describes the "what" of the system to be produced and not "how."In this phase, a large document called Software Requirement Specification (SRS) document is created which contained a detailed description of what the system will do in the common language.

2. Design Phase:

This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. It defines the

overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).

3. Implementation and unit testing:

During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.

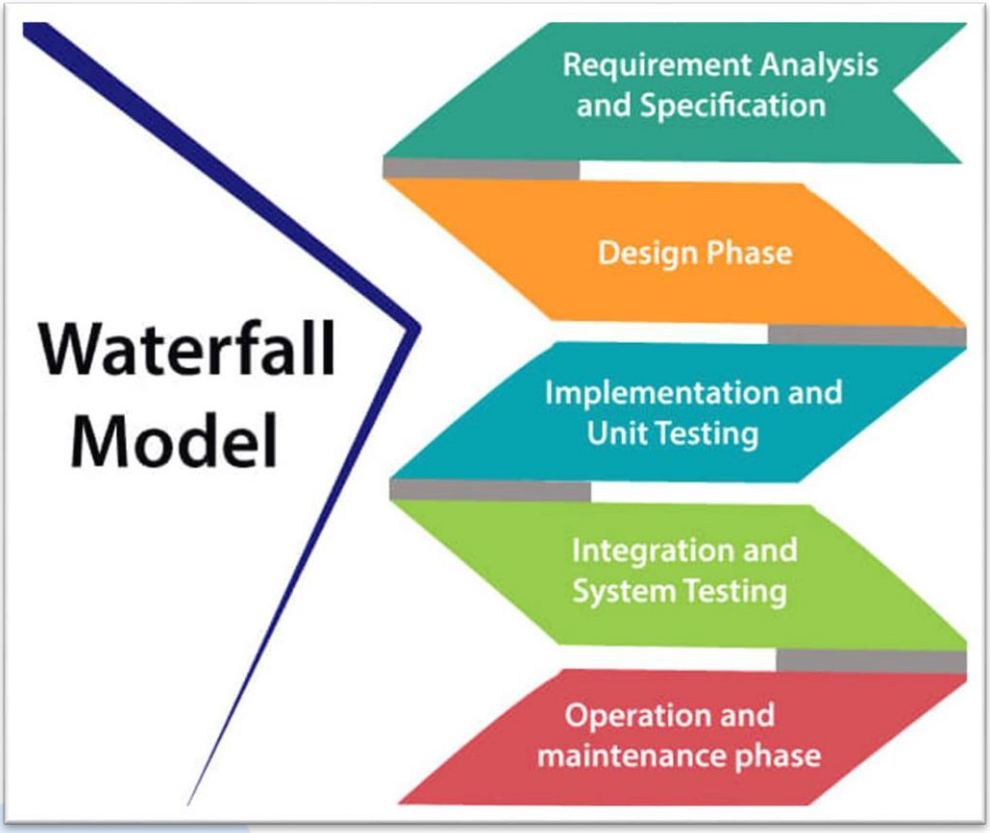
During testing, the code is thoroughly examined and modified. Small modules are tested in isolation initially. After that these modules are tested by writing some overhead code to check the interaction between these modules and the flow of intermediate output.

4. Integration and System Testing:

This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out. The better output will lead to satisfied customers, lower maintenance costs, and accurate results. Unit testing determines the efficiency of individual modules. However, in this phase, the modules are tested for their interactions with each other and with the system.

5. Operation and maintenance phase:

Maintenance is the task performed by every user once the software has been delivered to the customer, installed, and operational.



When to use SDLC Waterfall Model:-

Some Circumstances where the use of the Waterfall model is most suited are:

- When the requirements are constant and not changed regularly.
- A project is short
- The situation is calm
- Where the tools and technology used is consistent and is not changing
- When resources are well prepared and are available to use.

Advantages of Waterfall model:-

- This model is simple to implement also the number of resources that are required for it is minimal.
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
- The start and end points for each phase is fixed, which makes it easy to cover progress.

- The release date for the complete product, as well as its final cost, can be determined before development.
- It gives easy to control and clarity for the customer due to a strict reporting system.

Disadvantages of Waterfall model:-

- In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
- This model cannot accept the changes in requirements during development.
- It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement, It becomes tough to go back and change it.
- Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare.

2. Spiral Model:-

The spiral model, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.

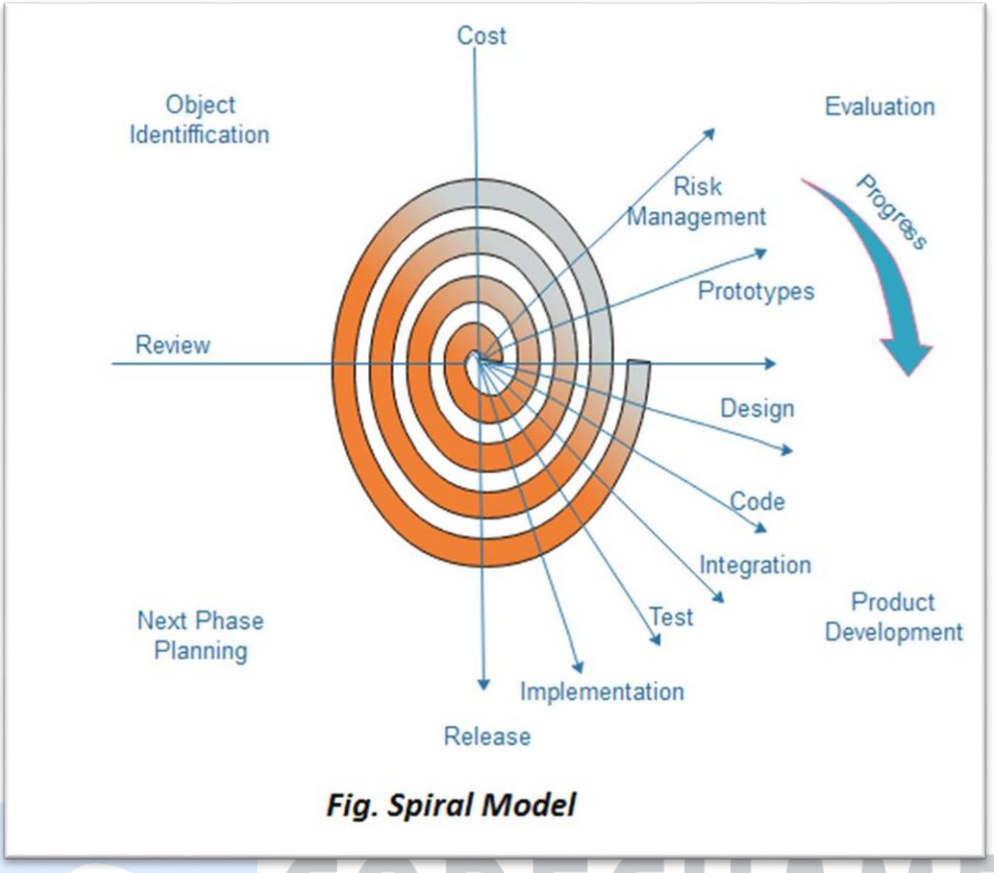


Fig. Spiral Model

Each cycle in the spiral is divided into four parts:

Objective setting: Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.

Risk Assessment and reduction: The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

Development and validation: The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.

Planning: Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks.

The **risk-driven** feature of the spiral model allows it to accommodate any mixture of a specification-oriented, prototype-oriented, simulation-oriented, or another type of approach. An essential element of the model is that each period of the spiral is completed by a review that includes all the products developed during that cycle, including plans for the next cycle. The spiral model works for development as well as enhancement projects.

When to use Spiral Model: -

- When deliverance is required to be frequent.
- When the project is large
- When requirements are unclear and complex
- When changes may require at any time
- Large and high budget projects

Advantages: -

- High amount of risk analysis
- Useful for large and mission-critical projects.

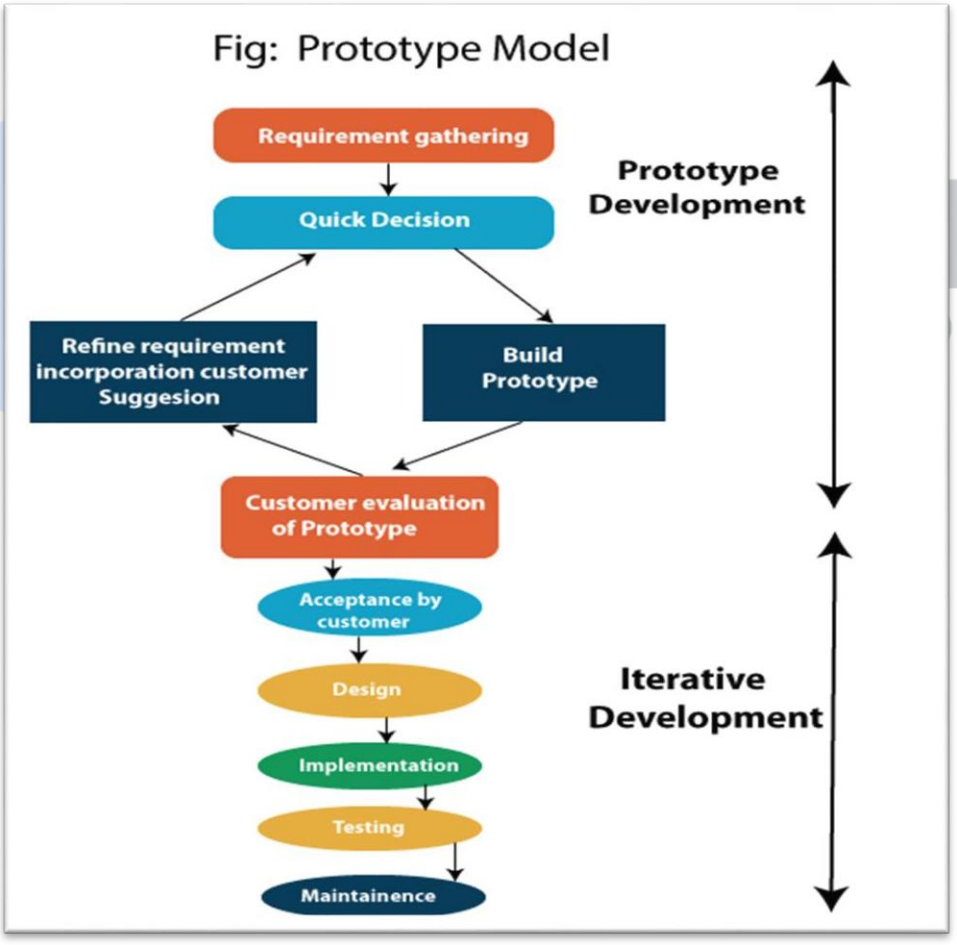
Disadvantages: -

- Can be a costly model to use.
- Risk analysis needed highly particular expertise
- Doesn't work well for smaller projects

3. Prototype Model:-

The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possible exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software.

In many instances, the client only has a general view of what is expected from the software product. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs, and the output requirement, the prototyping model may be employed.



Steps of Prototype Model:-

1. Requirement Gathering and Analyst
2. Quick Decision
3. Build a Prototype
4. Assessment or User Evaluation
5. Prototype Refinement
6. Engineer Product

Advantage of Prototype Model:-

- Reduce the risk of incorrect user requirement
- Good where requirement is changing/uncommitted
- Regular visible process aids management
- Support early product marketing
- Reduce Maintenance cost.
- Errors can be detected much earlier as the system is made side by side.

Disadvantage of Prototype Model:-

- An unstable/badly implemented prototype often becomes the final product.
- Require extensive customer collaboration
- Costs customer money
- Needs committed customer
- Difficult to finish if customer withdraw
- May be too customer specific, no broad market
- Difficult to know how long the project will last.
- Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
- Prototyping tools are expensive.
- Special tools & techniques are required to build a prototype.
- It is a time-consuming process.

Fourth Generation Techniques (4GT):-

- The term fourth generation techniques (4GT) encompasses a broad array of software tools that have one thing in common: each enables the software engineer to specify some characteristic of software at a high level.
- There is little debate that the higher the level at which software can be specified to a machine, the faster a program can be built.

- The 4GT paradigm for software engineering focuses on the ability to specify software using specialized language forms or a graphic notation that describes the problem to be solved in terms that the customer can understand.

4GT Tools:-

Currently, a software development environment that supports the 4GT paradigm includes some or all of the following tools:

1. Nonprocedural languages for database query
2. Report generation
3. Data manipulation
4. Screen interaction and definition
5. Code generation
6. High-level graphics capability
7. Spreadsheet capability
8. Automated generation of HTML
9. Similar languages used for Web-site creation using advanced software tools.

Requirements Gatherings:-

- 4GT begins with a requirements gathering step.

- Ideally, the customer would describe requirements and these would be directly translated into an operational prototype. But this is unworkable.
- The customer may be unsure of what is required, may be ambiguous in specifying facts that are known, and may be unable or unwilling to specify information in a manner that a 4GT tool can consume.

Design:-

- For small applications, it may be possible to move directly from the requirements gathering step to implementation using a nonprocedural fourth generation language (4GL) or a model composed of a network of graphical icons.
- However, for larger efforts, it is necessary to develop a design strategy for the system, even if a 4GL is to be used.
- The use of 4GT without design (for large projects) will cause the same difficulties (poor quality, poor maintainability, poor customer acceptance) that have been encountered when developing software using conventional approaches.

Implimentation:-

- Implementation using a 4GL enables the software developer to represent desired results in a manner that leads to automatic generation of code to create those results.

- Obviously, a data structure with relevant information must exist and be readily accessible by the 4GL.
- To transform a 4GT implementation into a product, the developer must conduct thorough testing, develop meaningful documentation, and perform all other solution integration activities that are required in other software engineering paradigms.
- In addition, the 4GT developed software must be built in a manner that enables maintenance to be performed expeditiously.

Advantages:-

- Development time is reduced, when used for small and intermediate applications.
- The interaction between user and developer help in detection of errors.
- When integrated with CASE tools and code generators, fourth generation techniques provide a solution to most of the software engineering problems.

Disadvantages:-

- Difficult to use.
- Limited to only small business information systems.

Metrics & Measurements: -

Software Metrics:-

- A software metric is a measure of software characteristics which are measurable or countable. Software metrics are valuable for many reasons, including measuring software performance, planning work items, measuring productivity, and many other uses.
- Within the software development process, many metrics are that are all connected. Software metrics are similar to the four functions of management: Planning, Organization, Control, or Improvement.

Classification of Software Metrics:-

Software metrics can be classified into two types as follows:

1. Product Metrics: These are the measures of various characteristics of the software product. The two important software characteristics are:

a) Size and complexity of software.
b) Quality and reliability of software.

2. Process Metrics: These are the measures of various characteristics of the software development process. For example, the efficiency of fault detection. They are used to measure the characteristics of methods, techniques, and tools that are used for developing software.

Types of Metrics:-

1. **Internal metrics:** Internal metrics are the metrics used for measuring properties that are viewed to be of greater importance to a software developer. For example, Lines of Code (LOC) measure.

2. **External metrics:** External metrics are the metrics used for measuring properties that are viewed to be of greater importance to the user, e.g., portability, reliability, functionality, usability, etc.
3. **Hybrid metrics:** Hybrid metrics are the metrics that combine product, process, and resource metrics. For example, cost per FP where FP stands for Function Point Metric.
4. **Project metrics:** Project metrics are the metrics used by the project manager to check the project's progress. Data from the past projects are used to collect various metrics, like time and cost; these estimates are used as a base of new software.

As the project proceeds, the project manager will check its progress from time-to-time and will compare the effort, cost, and time with the original effort, cost and time. Also understand that these metrics are used to decrease the development costs, time efforts and risks. The project quality can also be improved. As quality improves, the number of errors and time, as well as cost required, is also reduced.

Advantage of Software Metrics:-

- Comparative study of various design methodology of software systems.
- For analysis, comparison, and critical study of different programming language concerning their characteristics.
- In comparing and evaluating the capabilities and productivity of people involved in software development.
- In the preparation of software quality specifications.
- In the verification of compliance of software systems requirements and specifications.
- In making inference about the effort to be put in the design and development of the software systems.
- In getting an idea about the complexity of the code.

- In taking decisions regarding further division of a complex module is to be done or not.
- In guiding resource manager for their proper utilization.
- In comparison and making design tradeoffs between software development and maintenance cost.
- In providing feedback to software managers about the progress and quality during various phases of the software development life cycle.
- In the allocation of testing resources for testing the code.

Disadvantage of Software Metrics:-

- The application of software metrics is not always easy, and in some cases, it is difficult and costly.

- The verification and justification of software metrics are based on historical/empirical data whose validity is difficult to verify.
- These are useful for managing software products but not for evaluating the performance of the technical staff.
- The definition and derivation of Software metrics are usually based on assuming which are not standardized and may depend upon tools available and working environment.
- Most of the predictive models rely on estimates of certain variables which are often not known precisely.

Software Measurement:-

To assess the quality of the engineered product or system and to better understand the models that are created, some measures are used.

These measures are collected throughout the software development life cycle with an intention to improve the software process on a continuous basis.

Measurement helps in estimation, quality control, productivity assessment and project control throughout a software project. Also, measurement is used by software engineers to gain insight into the design and development of the work products.

In addition, measurement assists in strategic decision-making as a project proceeds.

Software measurements are of two categories, namely, direct measures and indirect measures.

1. **Direct measures** include software processes like cost and effort applied and products like lines of code produced, execution speed, and other defects that have been reported.
2. **Indirect measures** include products like functionality, quality, complexity, reliability, maintainability, and many more.

Generally, software measurement is considered as a management tool which if conducted in an effective manner, helps the project manager and the entire software team to take decisions that lead to successful completion of the project. Measurement process is characterized by a set of five activities, which are listed below:-

1. **Formulation:** This performs measurement and develops appropriate metric for software under consideration.
2. **Collection:** This collects data to derive the formulated metrics.
3. **Analysis**: This calculates metrics and the use of mathematical tools.
4. **Interpretation:** This analyzes the metrics to attain insight into the quality of representation.

5. **Feedback:** This communicates recommendation derived from product metrics to the software team.

collection and analysis activities drive the measurement process. In order to perform these activities effectively, it is recommended to automate data collection and analysis, establish guidelines and recommendations for each metric, and use statistical techniques to interrelate external quality features and internal product attributes.

Project Management:-

Project: - A project is a temporary effort to create a unique product, service or result. A project has a definite start and end.

Project management:- Project Management is not about managing people alone. PMI bifurcates project management into different process groups and knowledge areas. Process groups include initiating, planning, executing, monitoring and controlling, and closing.

Project Characteristics:-

A project is not normal day to day activity undertaken by organization rather it is specific, non-routine activity of varying time frame and impact viability of the business in the long run. A typical project has following characteristics:

1. **Timeline:** A project has a definite timeline with measurable starting and end point.
2. **Resources:** A project has limited resource of capital and manpower.
3. **Tools:** Special type of tools and techniques are used for project management (Gantt Charts, etc.)
4. **Team:** Project management requires diverse team stretching across departments and functions.

Project Life Cycle:-

A typical project is divided into following phases. Each phase of the project has its own importance and impact on overall success of the project.

1. **Initiation Phase:** In this phase of the project, feedback received from customers is analyzed and brainstorming is done as to develop new product or modify existing product to meet the new demands.
2. **Project Definition Phase:** In this phase of the project efforts are made to define the solution for the problem posed by customers.
3. **Feasibility Study:** In this phase, planning of the project is made and definite milestones are established.
4. **Project Execution:** In this phase all activities and milestones established in the earlier phase are executed in a timely and orderly manner. This phase utilizes maximum of all resources.
5. **Project Conclusion:** This is the last phase of the project. In this phase, final product or service is handed over to the operations team for commercial production.

Project Management Activities:-

Project management activities are mainly divided into three main categories Planning, Scheduling and Controlling.

1. **Planning:** Planning activities include defining project objective, resource planning, etc.
2. **Scheduling:** Scheduling activities include developing detailed milestones and guidelines for the project. These activities are performed typically before actual initiation of the project.
3. **Controlling:** Controlling activities include developing budget and finance control points, measuring of scheduled tasks are performed.

Project Management Techniques:-

There are several techniques utilized for project management. Some of the techniques are as follows, and they are mainly used for project scheduling.

1. **Gantt Charts:** These charts are used to depict the project tasks against time. It monitors progress of individual project tasks and also highlights dependency if any between those project tasks.
2. **Network Planning Techniques:** These techniques show the relationship between project activities, project duration, critical path, constraints of non-critical activities and resource utilization. There are two types of network planning techniques Critical Path Method (CPM) and Program Evaluation and Review Technique (PERT).