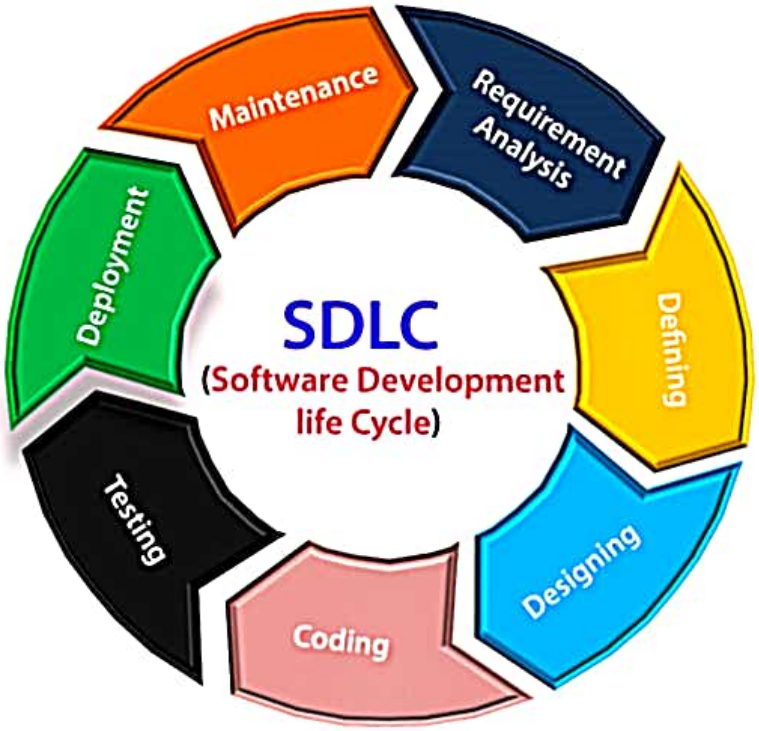
Software Development Life Cycle (SDLC)

Chap-01

What is SDLC?

- SDLC is a process followed for a software project, within a software organization.
- It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software.
- The life cycle defines a methodology for improving the quality of software and the overall development process.


Maintenance
Requirement Analysis
Deployment
Defining
SDLC
(Software Development life Cycle)
Testing
Designing
Coding

Stage 1: Planning and Requirement Analysis

- Requirement analysis is the most important and fundamental stage in SDLC.
- It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry.
- Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.

Stage2: Defining Requirements

- Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stakeholders.
- This is accomplished through "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project life cycle.

Stage3: Designing the Software

- The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project.
- This phase is the product of the last two, like inputs from the customer and requirement gathering.

Stage4: Developing the project

- In this phase of SDLC, the actual development begins, and the programming is built.
- The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.

Stage5: Testing

- After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.
- However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage6: Deployment

- Once the software is certified, and no bugs or errors are stated, then it is deployed.
- Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment.
- After the software is deployed, then its maintenance begins.

Stage7: Maintenance

- Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time.
- This procedure where the care is taken for the developed product is known as maintenance.

Software Requirements

Chap-02

Requirement Engineering Process

- It is a four step process, which includes –
  - Feasibility Study
  - Requirement Gathering
  - Software Requirement Specification
  - Software Requirement Validation

Feasibility study

- The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.
- The output of this phase should be a feasibility study report that should contain adequate comments and recommendations for management about whether or not the project should be undertaken

Requirement Gathering

- If the feasibility report is positive towards undertaking the project, next phase starts with gathering requirements from the user.
- Analysts and engineers communicate with the client and end-users to know their ideas on what the software should provide and which features they want the software to include.

Software Requirement Specification

- SRS is a document created by system analyst after the requirements are collected from various stakeholders.
- SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.

Software Requirement Validation

- After requirement specifications are developed, the requirements mentioned in this document are validated.
- Requirements can be checked against following conditions -
  - If they can be practically implemented
  - If they are valid and as per functionality and domain of software
  - If there are any ambiguities
  - If they are complete
  - If they can be demonstrated

Software Requirements Characteristics

- Gathering software requirements is the foundation of the entire software development project. Hence they must be clear, correct and well-defined.
- A complete Software Requirement Specifications must be:
  - Clear
  - Correct
  - Consistent
  - Reasonable
  - Compréhensible
  - Modifiable
  - Verifiable
  - Prioritized
  - Unambiguous

Software Requirements

- Functional Requirements
- Non-functional requirements
- User Requirements
- System Requirements
- Interface Specification

Functional Requirements

- These are the requirements that the end user specifically demands as basic facilities that the system should offer.
- All these functionalities need to be necessarily incorporated into the system as a part of the contract.
- These are represented or stated in the form of input to be given to the system, the operation performed and the output expected.
- They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Non-functional requirements

- These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.
- They basically deal with issues like:
  - Portability
  - Security
  - Maintainability
  - Reliability
  - Scalability
  - Performance
  - Reusability
  - Flexibility

Interface requirements

- UI is an important part of any software or hardware or hybrid system. A software is widely accepted if it is -
  - easy to operate
  - quick in response
  - effectively handling operational errors
  - providing simple yet consistent user interface
- User acceptance majorly depends upon how user can use the software. UI is the only way for users to perceive the system.
- A well performing software system must also be equipped with attractive, clear, consistent and responsive user interface. Otherwise the functionalities of software system can not be used in convenient way.

User Requirements

- The User Requirements Specification describes the business needs for what users require from the system.
- User Requirements Specifications are written early in the validation process, typically before the system is created.
- They are written by the system owner and end-users, with input from Quality Assurance.
- User requirements, often referred to as user needs, describe **what the user does** with the system, such as what activities that users must be able to perform.

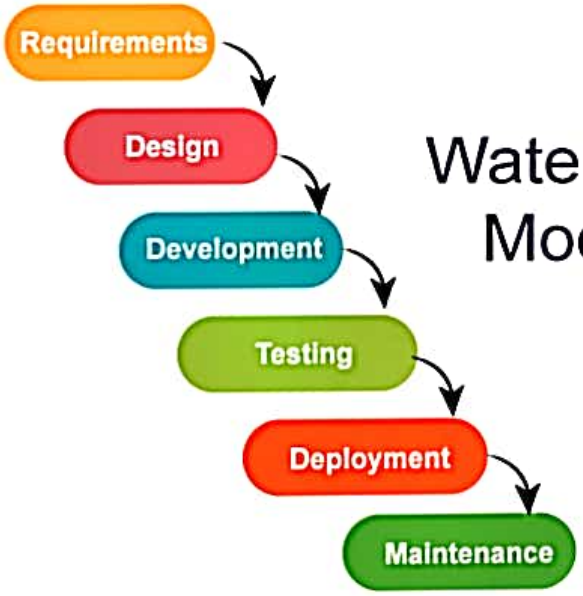
System Requirements

- A **System Requirements Specification (SRS)** (also known as a Software Requirements Specification) is a document or set of documentation that describes the features and behavior of a system or software application.
- It includes a variety of elements that attempts to define the intended functionality required by the customer to satisfy their different users.

Software Development Process Models.

- There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.
- Following are the most important and popular SDLC models followed in the industry −
  - Waterfall Model
  - Iterative Model
  - Prototyping model
  - RAD Model

Waterfall Model



- The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**.
- It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.
- The Waterfall model is the earliest SDLC approach that was used for software development.
- The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

- The sequential phases in Waterfall model are –
- **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment

Advantages

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.

Disadvantages

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.

Iterative Model

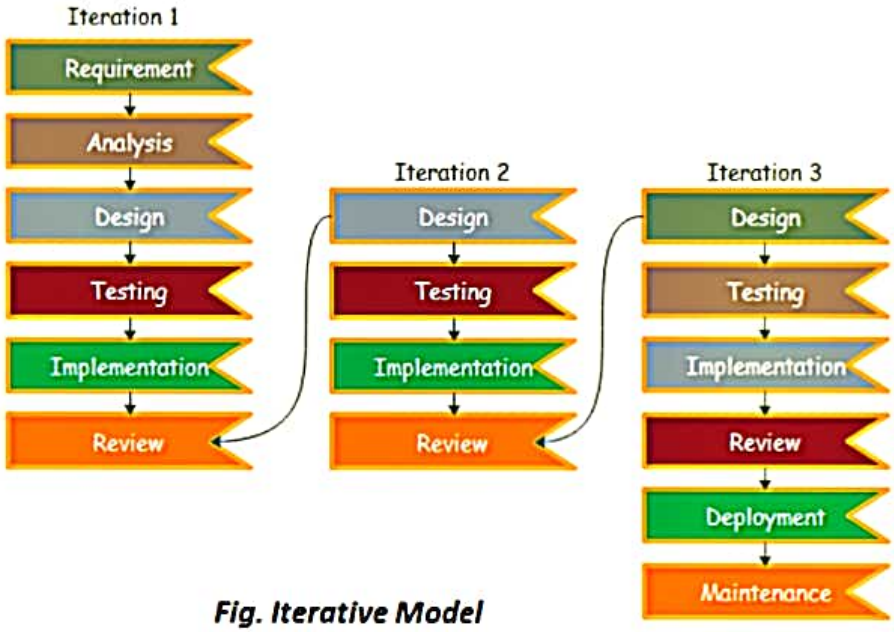


Fig. Iterative Model

- In this Model, you can start with some of the software specifications and develop the first version of the software.
- After the first version if there is a need to change the software, then a new version of the software is created with a new iteration.
- Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.
- At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative).

When to use the Iterative Model?

- When requirements are defined clearly and easy to understand.
- When the software application is large.
- When there is a requirement of changes in future.

Advantage

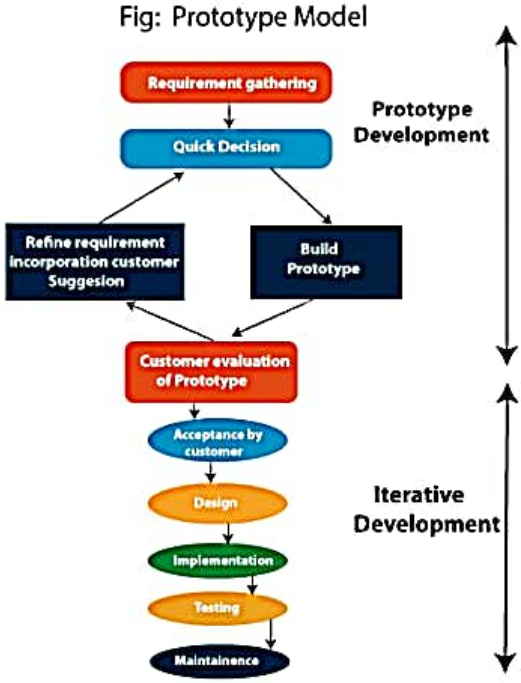
- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Testing and debugging during smaller iteration is easy.
- It is easily acceptable to ever-changing needs of the project.
- Risks are identified and resolved during iteration.
- Limited time spent on documentation and extra time on designing.

Disadvantage

- It is not suitable for smaller projects.
- More Resources may be required.
- Design can be changed again and again because of imperfect requirements.
- Requirement changes can cause over budget.
- Project completion date not confirmed because of changing requirements.

Prototype Model

Fig: Prototype Model



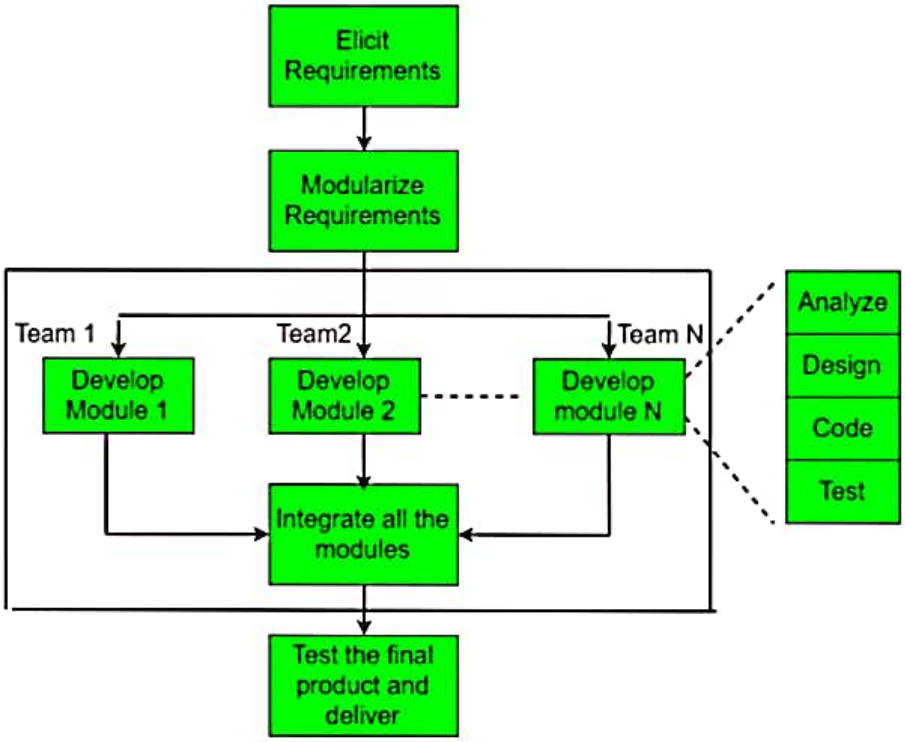
- The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system.
- In many instances, the client only has a general view of what is expected from the software product. In such 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
  - Requirement Gathering and Analyst
  - Quick Decision
  - Build a Prototype
  - Assessment or User Evaluation
  - Prototype Refinement
  - Engineer Product

- **Step 1: Requirements gathering and analysis**
  - A prototyping model starts with requirement analysis. In this phase, the requirements of the system are defined in detail. During the process, the users of the system are interviewed to know what is their expectation from the system.
- **Step 2: Quick design**
  - The second phase is a preliminary design or a quick design. In this stage, a simple design of the system is created. However, it is not a complete design. It gives a brief idea of the system to the user. The quick design helps in developing the prototype.
- **Step 3: Build a Prototype**
  - In this phase, an actual prototype is designed based on the information gathered from quick design. It is a small working model of the required system.

- **Step 4: Initial user evaluation**
  - In this stage, the proposed system is presented to the client for an initial evaluation. It helps to find out the strength and weakness of the working model. Comment and suggestion are collected from the customer and provided to the developer.
- **Step 5: Refining prototype**
  - If the user is not happy with the current prototype, you need to refine the prototype according to the user's feedback and suggestions.
  - This phase will not over until all the requirements specified by the user are met. Once the user is satisfied with the developed prototype, a final system is developed based on the approved final prototype.

RAD (Rapid Application Development)

- The **RAD (Rapid Application Development)** model is based on prototyping and iterative development with no specific planning involved.
- If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.
- A software project can be implemented using this model if the project can be broken down into small modules wherein each module can be assigned independently to separate teams. These modules can finally be combined to form the final product.
- Development of each module involves the various basic steps as in waterfall model i.e analyzing, designing, coding and then testing, etc.
- Another striking feature of this model is a short time span i.e the time frame for delivery(time-box) is generally 60-90 days.


Elicit Requirements
Modularize Requirements
Team 1
Team2
Team N
Develop Module 1
Develop Module 2
Develop module N
Analyze
Design
Code
Test
Integrate all the modules
Test the final product and deliver

- This model consists of 4 basic phases:
- **Requirements Planning**
  - It involves the use of various techniques used in requirements elicitation(**the practice of researching and discovering the requirements of a system from users, customers, and other stakeholders**).
  - It also consists of the entire structured plan describing the critical data, methods to obtain it and then processing it to form final refined model.
- **User Description**
  - This phase consists of taking user feedback and building the prototype using developer tools. In other words, it includes re-examination and validation of the data collected in the first phase. The dataset attributes are also identified and elucidated in this phase.

- **Construction**
  - In this phase, refinement of the prototype and delivery takes place. It includes the actual use of powerful automated tools to transform process and data models into the final working product. All the required modifications and enhancements are too done in this phase.
- **Cutover**
  - All the interfaces between the independent modules developed by separate teams have to be tested properly. The use of powerfully automated tools and subparts makes testing easier. This is followed by acceptance testing by the user.

- **Advantages**
  - Use of reusable components helps to reduce the cycle time of the project.
  - Feedback from the customer is available at initial stages.
  - Reduced costs as fewer developers are required.
  - Use of powerful development tools results in better quality products in comparatively shorter time spans.
- **Disadvantages**
  - The use of powerful and efficient tools requires highly skilled professionals.
  - The absence of reusable components can lead to failure of the project.
  - The team leader must work closely with the developers and customers to close the project in time.
  - Customer involvement is required throughout the life cycle.
  - It is not meant for small scale projects as for such cases, the cost of using automated tools and techniques may exceed the entire budget of the project.

What makes RUP Special?

The Specialty of the RUP is it reduces the unexpected development cost and prevent wastage of resources.

Phases of RUP

- There are five phases of RUP

1. **Inception:**
    1. Communication and planning.
    2. Identification of Project Scope.
    3. Customer requirement identification
    4. Project Plan, Project goal, Risk identifications are made and identified.

2. **Elaboration:**
    1. Elaboration means describing something in more details.
    2. Here we go into more detail of the 1st phase.
    3. Redefine if we feel need, cancel project as well if needed.

Phases of RUP

- There are five phases of RUP

3. **Construction:**
    1. Here we develop and complete the project based on the data we get from previous stages.
    2. Coding of project is done here.
    3. All kind of testing are also done here.

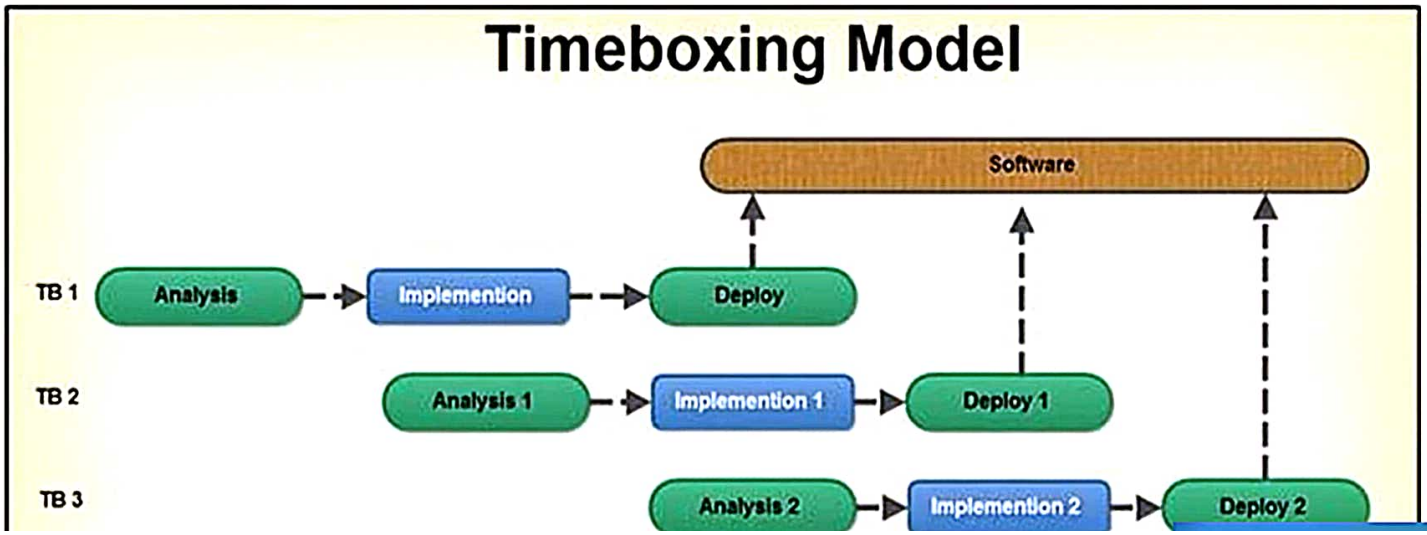
4. **Transition:**
    1. Here finally project is transit from development environment to production.
    2. Here we also set the project on beta testing mode.
    3. Remove the bugs from project based on customer feedback.

Phases of RUP

- There are five phases of RUP

5. **Production:**
    1. This is final phase of the model.
    2. Project is maintained here.
    3. Project is updated here accordingly.

Timeboxing Model



Timeboxing Model

- In time boxing model, development is done iteratively as in the iterative enhancement model.
- In time boxing model, each iteration is done in a timebox of fixed duration.
- The functionality to be developed is adjusted to fit the duration of the timebox.
- Moreover, each timebox is divided into a sequence of fixed stages where each stage performs a clearly defined task (analysis, implementation, and deploy) that can be done independently.
- This model also requires that the time duration of each stage is approximately equal so that pipelining concept is employed to have the reduction in development time and product releases.

Timeboxing

- Time boxing is an Iterative development but
  - fix an iteration duration, then determine the specifications
- Timeboxing model divide iteration in a few equal stages
- Use pipelining concepts to execute iterations in parallel

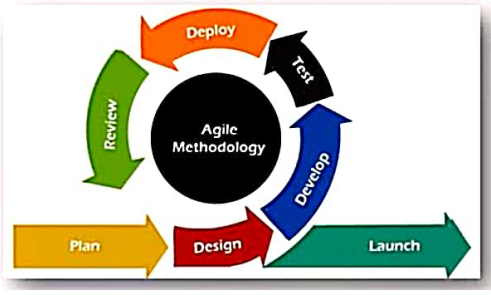
About Agile Model

➢ **About Agile:**

- Mostly used model in todays digital era.
- Agile means "The ability to respond to changes from requirements, technology & people"
- It is an incremental and iterative process of software development.

➢ **Working with Example:**

- Divides requirements into multiple iterations & provide specific functionality for the release.
- Delivers multiple software requirements.
- Each iterations are lasts from two to three weeks.
- Direct collaboration with customers.
- Rapid project development.



XP – Extreme Programming

- It is the type of Agile Development we develop software with small team where the software requirements changes rapidly.

- XP is summed up or based on twelve Practices. . .

XP Practices

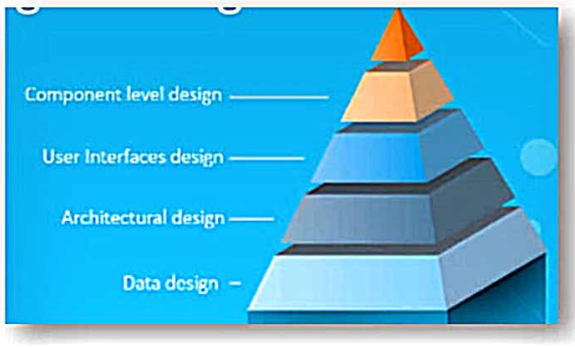
- *Faster Feedback*
  - *Test Driven Development*
  - *On-site customer*
  - *Pair Programming*
- *Continuous Process*
  - *Continuous Integration*
  - *Refactoring*
  - *Short Releases*
- *Shared Understanding*
  - *The Planning Game*
  - *Simple Design*
  - *System Metaphor*
  - *Collective Ownership*
  - *Coding Standard*
- *Developers Welfare*
  - *40 hour workweek*

Design Models

- The design phase of software development, transforming the customer requirements as described in the SRS documents into design forms.
- Designing a model is an important phase and is a multi-process that represent the data structure, program structure, interface characteristic and procedural details.

➢**There are four types of design elements / models**

1. Data Design Element / Model
2. User Interface Design Element / Model
3. Architectural Design Element / Model
4. Component Level Design Element / Model



Architectural Design Model

- Architecture Design Model serves as a blueprint for a system.
- The architecture focuses on the early design decisions.

Architectural Design Styles describe:

➢Set of hardware & software components that will perform a function required by the system.
Eg. Database, Modules, Frameworks etc.

➢Set of connectors will help in coordination & communication between the components.

➢Conditions that how components can be integrated to form the system.

➢Semantic models that help the designer to understand the overall properties of the system.

Importance of Software Architecture

1. **Security:** The system is secured against malicious users by encryption or any other security measures due to layered software architecture.
2. **Performance:** It handle request and response of the page in minimum time.
3. **Maintainability:** Architectural design process uses easily modifiable and replaceable components which is easy to change them over time according to the new requirements.
4. **Safety:** Avoid critical functionalities in small components & improve communication of the system.
5. **Availability:** Architectural design process includes corresponding components, functionalities for handling the occurrence of any type of errors.

Decisions of Architectural Design

- The architectural design process differs as the system differs depending upon the type of system being developed.

➢**There are some common decisions that should be taken care of in any design process.**

❑How can the system be distributed across the network?

❑Which approach can be used to structure the system?

❑Which architectural styles are suitable for the proposed system?

❑How can software architecture be documented?

❑How can the system be decomposed into modules?

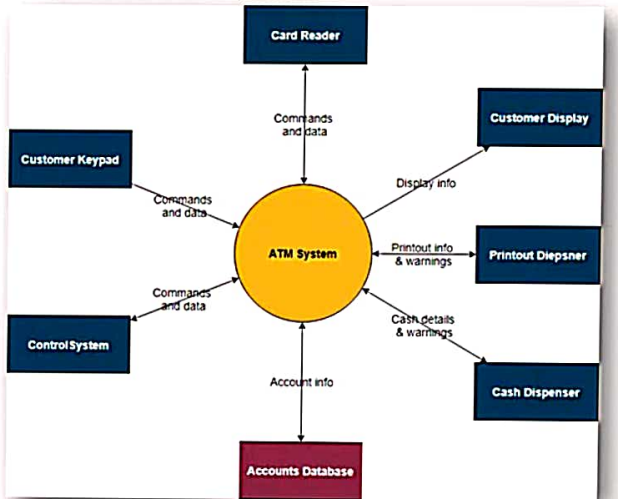
❑What control strategy must be used to control the operation of the components in the system?

❑How can architectural design be analyzed?

Taxonomy of Architectural styles

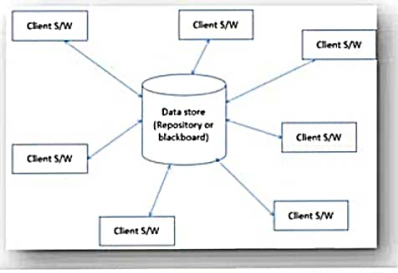
➢ **Architectural styles is establish a complete structure & components of all over the system.**

1. Data Centered Architecture
2. Data Flow Architecture
3. Object Oriented Architecture
4. Layered Architecture



Data Centered Architecture

- Data store at the center of this architecture and is accessed frequently by everyone.
- Update, add, delete or modify the data present within the store.
- It is widely used in DBMS, Library Information System etc.

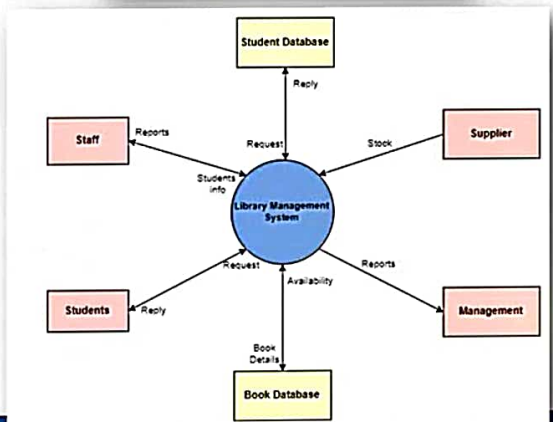


Advantages:

1. Repository of data is independent of clients
2. It may be simple to add additional clients.
3. Modification can be very easy.

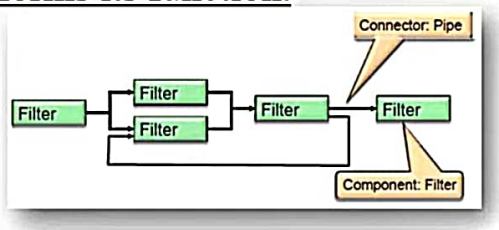
Disadvantages:

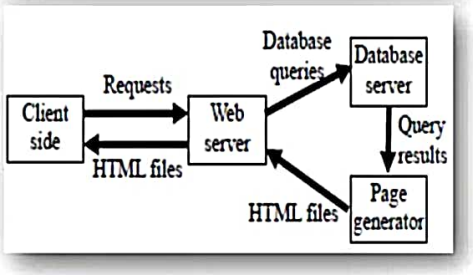
1. Data replication or duplication is possible.
2. Changes in data structure highly affect the clients.



Data Flow Architecture

- This architecture is used when input data to be transformed into output data through a series of computational manipulative components.
- **Pipe** is a connector which passes the data directionally from one filter to the next.
- **Filter** is a component reads the data from its input pipes and performs its function.
- This data and places the result on all output pipes.





Advantages:

1. With this design, concurrent execution is supported.

Disadvantages:

1. Does not allow greater user engagement.

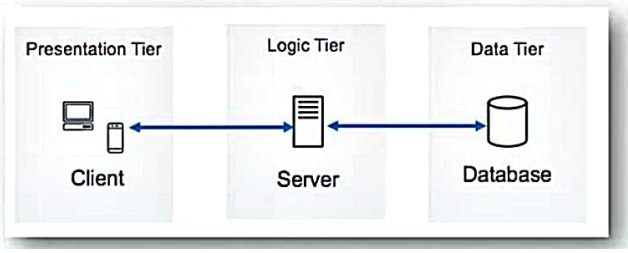
Object Oriented Architecture

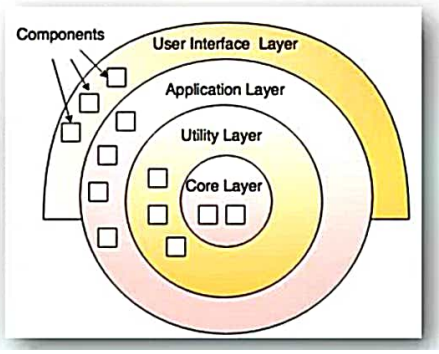
- Objects are the foundational building blocks for all kinds of software applications.

1. **Object:** Object is an instance of a class. Example Student S, Person;
2. **Class:** It defines all attributes, methods, which represents the functionality of the object.
3. **Encapsulation:** It is the process of binding similar types of elements of an abstraction.
4. **Abstraction:** It is the removal of irrelevant essentials from users.
5. **Inheritance:** It deriving a new class from existing one. It increases code reusability.
6. **Polymorphism:** It has multiple forms. Ex: draw graphic objects circle, rectangle, triangle
7. **Message Passing:** Sending and receiving data among objects through function parameters.

Layered Architecture

- Data moves from one level to another level for processing is called layered architecture.
- Number of different layers are defined every layer performing well-defined set of operations.
- **Outer layer** components manage the user interface operations.
- **Inner layer** components will perform the operating system interfacing.
- **Intermediate layers** provide utility services and application software functions.
- **Example:** E-commerce web applications development like Amazon.





Architectural Views

- It is generally used to represent entire architecture that is useful and meaningful to one or more stakeholders in system.

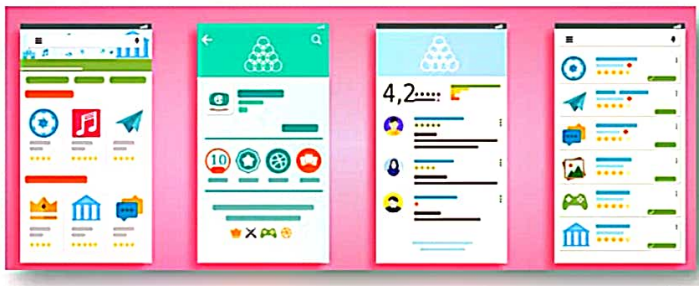
1. **Use case View:** Users view to handling product as per requirement. Use case diagram used.
2. **Design View:** Organize design information, significant features, entities and attributes.
3. **Process View:** Describe communication, behavior & synchronization aspects of the design.
4. **Implementation View:** It address source code integrators and developers of project.
5. **Deployment View:** It describes & explains environment within system runner & executed.

User Interface Design Model

- User interface is the front-end application view to which user interacts with the software.
- It determines how commands are given to the computer or the program and how data is displayed on the screen.

➢**The software becomes more popular if its user interface is:**

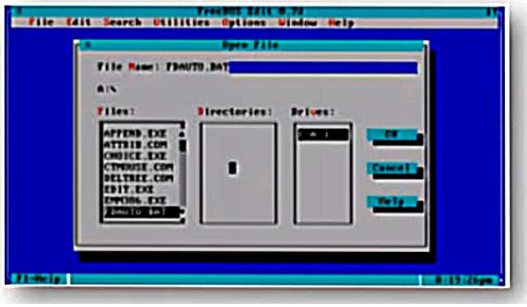
1. Attractive
2. Simple to use
3. Responsive in short time
4. Clear to understand
5. Consistent on all interfacing screens



Types of User Interface

Type 1: Text-Based User Interface OR Command Line Interface

- Text based interface primarily used keyboard handling data.
- Command Line Interface provide command prompt or coding tools where the user types the command towards the system.
- The user needs to remember the syntax of command and its use.
- Command Line Interface used by Technical people or Programmers.

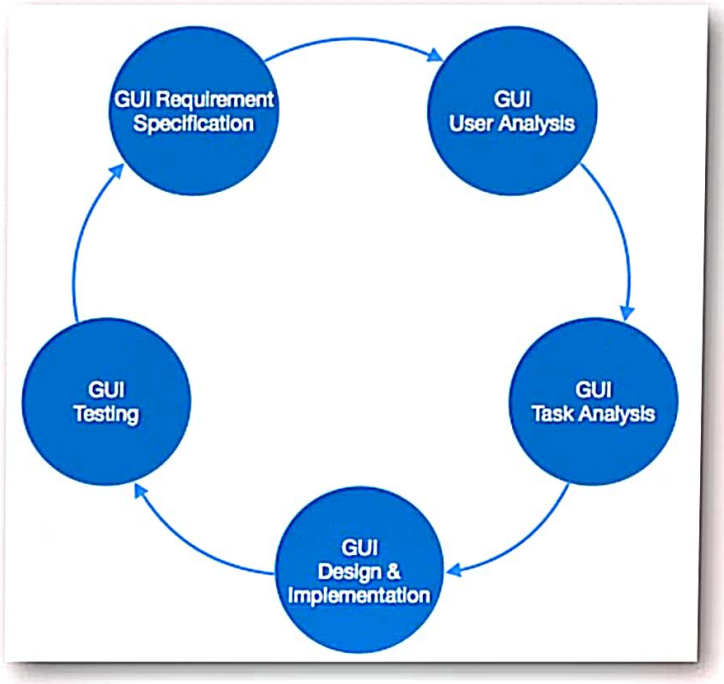


Types of User Interface

Type 2: Graphical User Interface

- Graphical User Interface provides the simple interactive interface to interact with the system.
- GUI can be a combination of both hardware and software.
- Graphical user interfaces are easy to learn as compared to the command-line interface.
- GUI provides multiple windows to the user simultaneously to interact with the system.

User Interface Design Process



User Interface Design Process

1. **GUI Requirement Gathering:** Designer analyze all functional & non functional requirements mentioned & discussed with customers.
2. **User Analysis:** Designer studies who is going to use the software GUI. The target audience matters as the design details change according to the knowledge level of the user.
3. **Task Analysis:** Designer analyze tasks, sub tasks & flow of the system.
4. **GUI Design & Implementation:** Designer generate actual GUI design by using different implementation tools like Wavemaker, Visual Studio etc.
5. **Testing:** GUI Testing done by Inner designer team, Organization & different stockholders in project.

User Interface Design Principles

1. **User Familiarity:** The interface should be based of user oriented terms & concepts.

   Example: In Windows OS, Terms like Desktop, Document, Folder, Rename, Power etc.

2. **Consistency:** The system command and menus should have the same format & parameter.

   Example: Group of fonts, sizes, colors etc.

3. **Minimal Surprise:** Users never like to see the system working in an unexpected manner.

   They get frustrated in such cases. Develop as per the user tendency.

   Example: Group of similar type of features , In MS Words, File -> New, Open, Save, Print, Close , Restart system

User Interface Design Principles

4. **Recoverability:** Users most often make mistakes while working with the system. These mistakes can be reduced but cannot be completely eliminated.

   Example: Undo, Rename

5. **User Diversity:** The interface should provide appropriate interaction facilities for the various types of the system user.

   Example: Larger Font, Privacy Settings etc.

6. **User Guidance:** The user interface should provide meaningful feedback when errors occur and provide different levels of help and advice.

   Example: User Manual, Forgot Password etc.

User Interface Design Golden Rules

1. Place the user in control.

✓The user should be able to easily enter and exit the mode.

✓Provide flexible interaction using keyboard, mouse or touch screen etc.

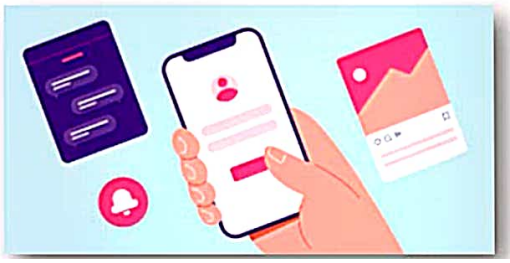
✓User must be able to interrupt the sequence to do some other work.

✓Display descriptive messages and text.

✓ Allow users to customize the interface like short cut keys etc.

✓Hide technical internal details from users.

✓Allow users to directly manipulate interface objects.



User Interface Design Golden Rules

2. Reduce Users' Memory Load

✓Reduce demand on short-term memory.

✓New features updated in system.

✓Define shortcuts that are intuitive.

✓The visual layout of the interface should be based on a real-world metaphor.

✓Disclose information in a hierarchical & progressive fashion.

3. Make the Interface Consistent

✓Meaningful user tasks performed.

✓Maintain consistency across a family of applications.

✓Keep interaction results the same as per user expectations.

Visually Consistent UI Elements

User Interface Design Issues

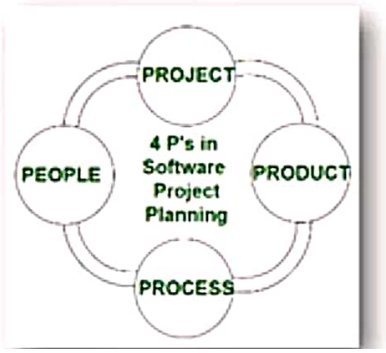
1. **Response Time:** Time between request and response of the system. If the response time is too long, then the user becomes frustrated.
2. **Error Handling:** Poor error message may result in rejecting the product.
3. **Help Facilities:** User requires help when he needs some information and he cannot find it then the user is in trouble.
4. **Application Accessibility:** It states whether the application is simple to interact with or not. Special guidelines are given to user while interacting with software.

The Management Spectrum

- In software engineering, the management spectrum describes the management of a software project.
- For properly building a product, there's a very important concept handle by Project Manager.
- These components play a very important role in your project that can help your team meet its goals and objective

➢**Effective software project management focuses on the four P's:**

1. People
2. Product
3. Process
4. Project.



1. People

- The most important contribution in software project is not made by system or tool, It is done by people. i.e. Human Resources.
- The success of project depend on selecting right kind of people with right talent.

➢**Depending on there roles & responsibilities, following are the main categories:**

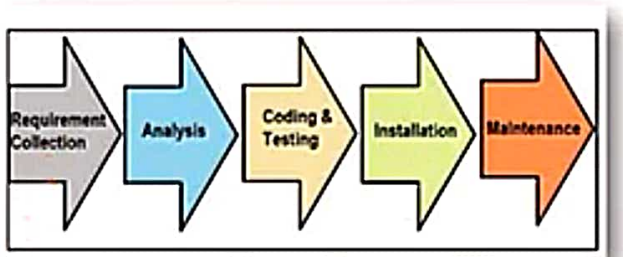
1. **Senior Manager:** Define the business issues & have significant influence on the project.
2. **Project Manager:** Plan, Motivate, Organize & Control the project work. Has problem solving skills & manageable team abilities.
3. **Software Engineer:** Who deliver technical skills which are necessary in project.
4. **Customer:** Who specify the requirement for the software project.
5. **End Users:** Who interact with the software product.

2. Product

- The product is the ultimate goal of the project.
- This is any types of software product that has to be developed or deliverable.
- Before a product can be planned, its objectives and scope should be established, alternate solutions should also be considered & technical & management constraints should be identified.
- Lack of these information, it is impossible to define reasonable and accurate estimation of the cost, identify possible risk & define manageable project schedule.

3. Process

- Project managers and team members should have a methodology and plan that complete project as per customer requirements.
- Without a clearly defined process, team members will not know what to do and when to carry out project activities.
- Using the right process will increase the project execution success rate that meets its original goals and objectives.
- The Process has several steps involved like, Documentation phase, Designing phase, Implementation phase, software configuration management, Deployment phase and Interaction phase.



4. Project

- The project is the complete software project that includes requirement analysis, development, delivery, maintenance and updates.
- The project manager plays a critical role for completing project.
- They are responsible to guide the team members to achieve the project's target and objectives.
- They helping & assisting them with issues, checking on cost and budget and making sure that the project stays on track with the given deadlines.
- They manage complete project activities to avoid project failure.

About Project Planning

- Before starting a software project, it is essential to decide which tasks to be performed and how to manage all tasks involved in the software development.

✓Project planning is an organized process from requirement gathering to testing and support.

✓Focuses on all activities required for successful completion of the project.

✓Prevents obstacles that arise in the project such as changes in projects, organization's objectives, non-availability of resources and so on.

✓Helps in better utilization of resources and optimal usage of the allotted time for a project.

✓Defines roles and responsibilities of the project management team members.

Software Project Manager

➢**Managing People**

- Act as Project Leader.
- Contact with all stakeholders.
- Managing human resources.

➢**Managing Project**

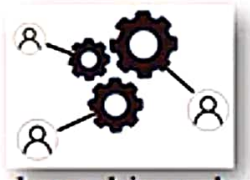
- Defining and setting up project scope.
- Managing project management activities.
- Monitoring progress and performance.
- Risk analysis at every phase.
- Take necessary step to avoid or come out of problems.

Project Planning Process

Project Planning Process

1. Identify Stakeholders Need:

- A project must meet the expectations of stakeholders & meet all of their needs to be effective.
- To consider their needs a project planner might identify the characteristics and qualities of the project at hand.



2. Identify Project Objectives:

- Project objectives must be specific, measurable, achievable, realistic and time-bound in order to be used to validate project success.
- All team members must contribute ideas and compromise on collective goals for the project.

Project Planning Process

3. Identify Deliverables & Due Date:

- A due date is a scheduled fixed date and time that an objective is due within a project.
- Deliverables are defined as the products, services or results that project quality, size, length, or any other standard that proves to be important in the success of project.

4. Identify Project Schedules:

- This stage is created through different tasks that label a project's start and end date.
- Schedules are used to make sure projects stay on task and are completed in a timely manner.

Project Planning Process

5. Provide Roles & Responsibilities:

- This stage allows for effective communication between everyone involved in the project.
- This segment includes identifying who is involved and what their individual tasks are.
- Everyone understand expected objectives, goals & output of project.

6. Identify Project Budget:

- Project budget are calculating the anticipated cost, changing budget cost and monitoring the budget.
- Therefore, it is crucial to prepare for unexpected alterations of project costs.

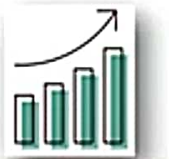
Project Planning Process

7. Identify Communication Plan:

- A communication plan is a tool that able to effectively communicate about a project to your client, team and other stakeholders.
- Communication plan has clear guidelines about how information will be shared & who is responsible within the project.

8. Provide Tracking & Management:

- It is an effective way to deliver projects on time and organize tasks.
- Features of this element include planning/scheduling, collaboration, documentation and evaluation.
- Management tools are track productivity and growth of project.



Recourses Used in Project Development

- Project resources simply mean resources that are required for successful development and completion of project on time and on budget.

Type 1: Human Recourses

- In software industry, manager, software developer, software testing and so on.
- These positions are according to their skills and specialty.

Resource Pyramid

Type 2: Reusable Components

- Managing budget for project is one of most important tasks that all project managers.
- The reusable resources are very helpful as they help in reducing overall cost of development.

Type 3: Hardware and Software tools

- It should be planned before starting development of project otherwise it way causes problems.
- These are actually material resources that are part of project.

Issues in Project Management

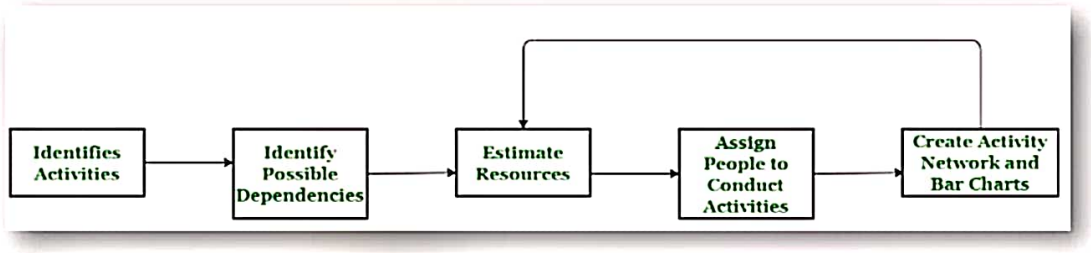
- Project is not completed on schedule.
- Changing customer requirements affect on schedule.
- Technical difficulties are generate.
- Miscommunication among project management.
- Essential software & hardware may be delivered late.
- In large project, Software engineer perform multiple tasks parallel.
- Tasks interdependencies are in project.
- Risk is not considered at beginning of project.

About Project Scheduling

- Project scheduling is responsible activity of project manager.
- Project schedule is a mechanism that is used to communicate and know about that tasks are needed and has to be performed in project.
- Project manager separate total work task in project into different activities. i.e. WBS
- Project Manager estimate time & recourses required to complete activities & organize them into coherent sequence.
- Effective project scheduling leads to success of project, reduced cost and increased customer satisfaction.

Project Scheduling Process

1. Identify all the functions / modules required to complete the project.
2. Break down large functions into small activities i.e. Develop WBS structure.
3. Determine the dependency among various activities.
4. Allocate resources to activities.
5. Assign people to conduct different activities.
6. Plan the beginning and ending dates for different activities.
7. Create Activity Network & Bar or Gantt charts.



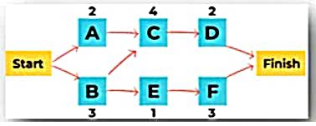
Project Scheduling Techniques

1. Critical Path Method:

- It help you determine both longest and shortest possible time take to complete a project.
- There are three essential elements: The tasks required to complete the project, Which tasks depend on the completion of others, A time estimate for each activity

Example:

- There are four tasks in the project – A, B, C, and D.
- Task B and D can only begin after task A completes, whereas task C has no such restriction.
- Task A will be time-sensitive as any delay in its completion can delay in task B & D.
- Called as Critical Task.
- This helps in identifying and separating the independent variables.
- Finally, it adds milestones to the project.



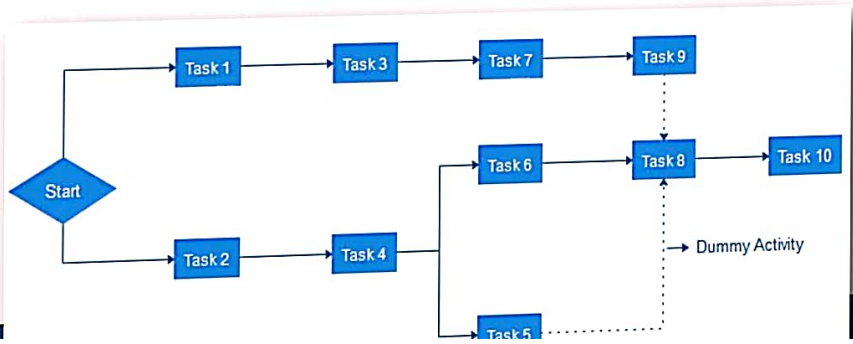
Project Scheduling Techniques

2. Program Evaluation and Review Technique (PERT):

- It is a way to schedule flow of tasks in a project and estimate total time taken to complete it.
- PERT charts offer a visual representation of the major activities (and dependencies) in a project

It calculate:

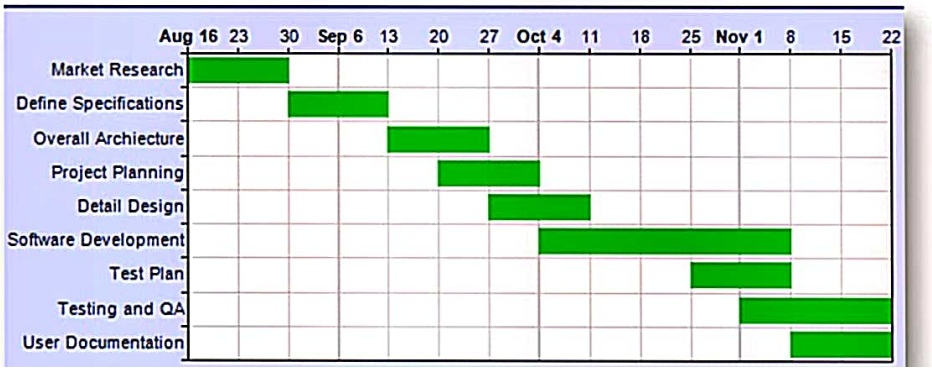
- **Optimistic time (O):** Quickest time you can complete a project
- **Pessimistic time (P):** Longest time it'll take to complete your project
- **Most likely time (M):** How long it'll take to finish your project if there are no problems.
- (O + 4M + P)/6



Project Scheduling Techniques

3. Gantt Chart:

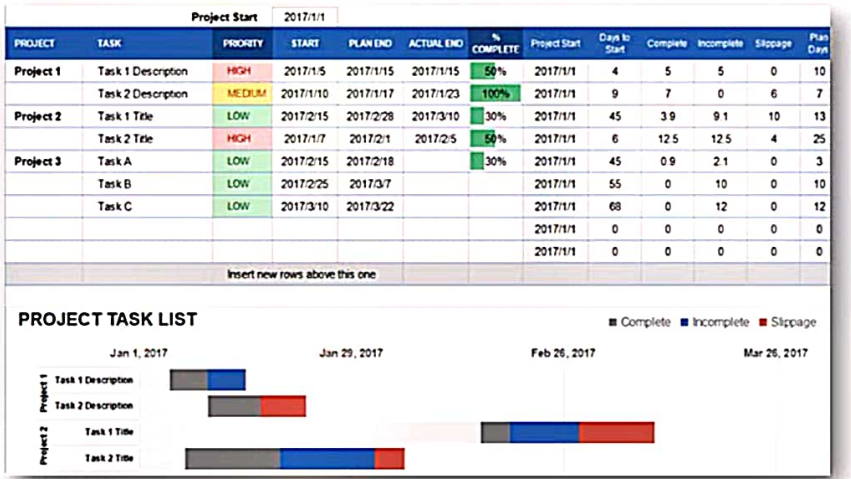
- A Gantt chart is a type of bar graph that project managers use for planning and scheduling in complex project.
- It represent each task horizontally on a bar chart, which shows the start and end dates & they frequently include deadlines & dependencies of tasks.
- It easier to visualize the progress of a project and see how different tasks interact with one another.



Project Scheduling Techniques

4. Task List:

- One of the simplest project scheduling techniques is the creation of a task list.
- Create task list using a word processor or spreadsheet software.
- It create a list of tasks and include important information like the task manager, start date, deadline & completion status.



Project Start 2017/1/1

PROJECT	TASK	PRIORITY	START	PLAN END	ACTUAL END	% COMPLETE	Project Start	Days to Start	Complete	Incomplete	Slippage	Plan Days
Project 1	Task 1 Description	HIGH	2017/1/5	2017/1/15	2017/1/15	50%	2017/1/1	4	5	5	0	10
	Task 2 Description	MEDIUM	2017/1/10	2017/1/17	2017/1/23	100%	2017/1/1	9	7	0	6	7
Project 2	Task 1 Title	LOW	2017/2/15	2017/2/28	2017/3/10	30%	2017/1/1	45	3.9	9.1	10	13
	Task 2 Title	HIGH	2017/1/7	2017/2/1	2017/2/5	50%	2017/1/1	6	12.5	12.5	4	25
Project 3	Task A	LOW	2017/2/15	2017/2/18		30%	2017/1/1	45	0.9	2.1	0	3
	Task B	LOW	2017/2/25	2017/3/7			2017/1/1	55	0	10	0	10
	Task C	LOW	2017/3/10	2017/3/22			2017/1/1	68	0	12	0	12
							2017/1/1	0	0	0	0	0
							2017/1/1	0	0	0	0	0
	Insert new rows above this one											

Project Scheduling Techniques

5. Fast Tracking :

- In Fast Tracking, Project is being implemented by either simultaneously executing many tasks or by overlapping many tasks to each other.
- **Example:** In software development project, designing and development can be taken up in parallel. Once design of essential features is ready and approved, development team can work on it. Meanwhile, the designing team will work on the remaining elements and functions.

6. Crashing:

- Crashing deals with involving more resources to finish the project on time.
- **Example:** Add more developer in project, Paying overtime to employee,
- Crashing can only be applied when it fits your project budget.