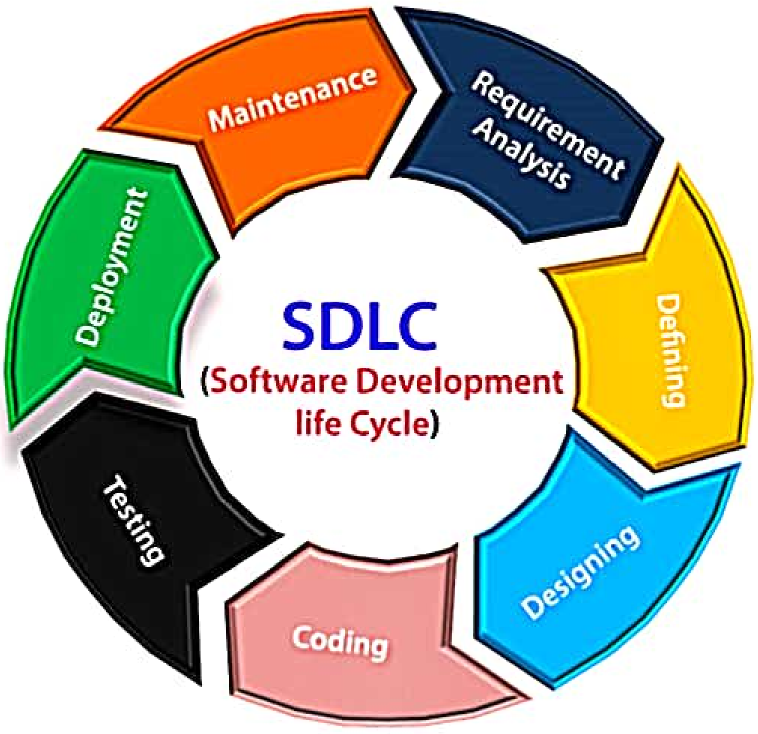
Software Development Life Cycle (SDLC)

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.


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

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.

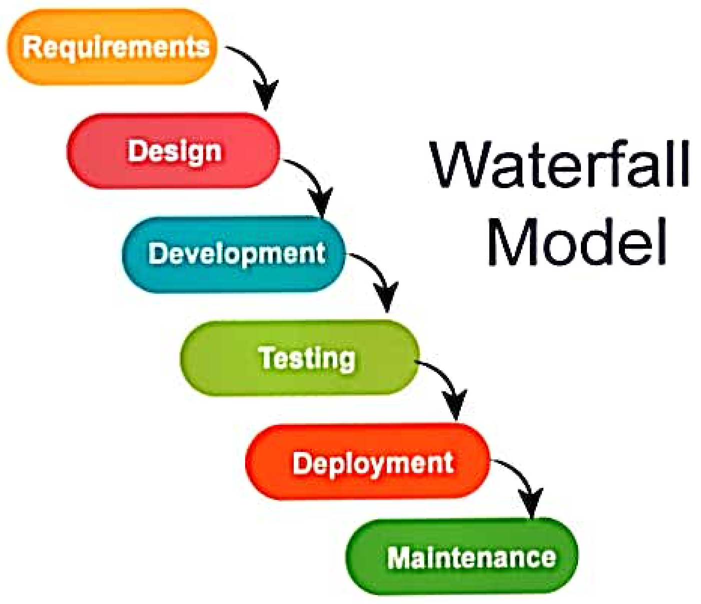
System Requirements

- 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

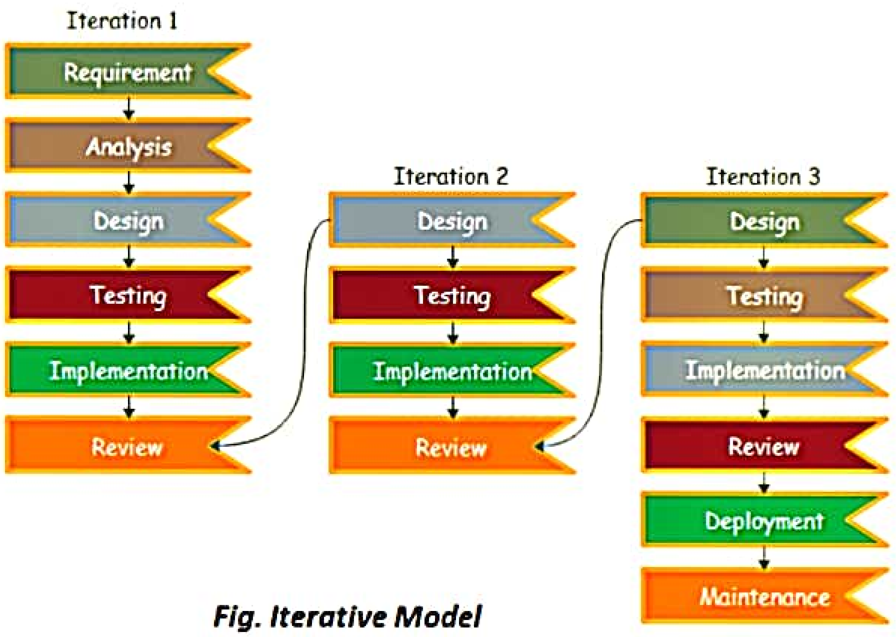


Fig. Iterative Model

- 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

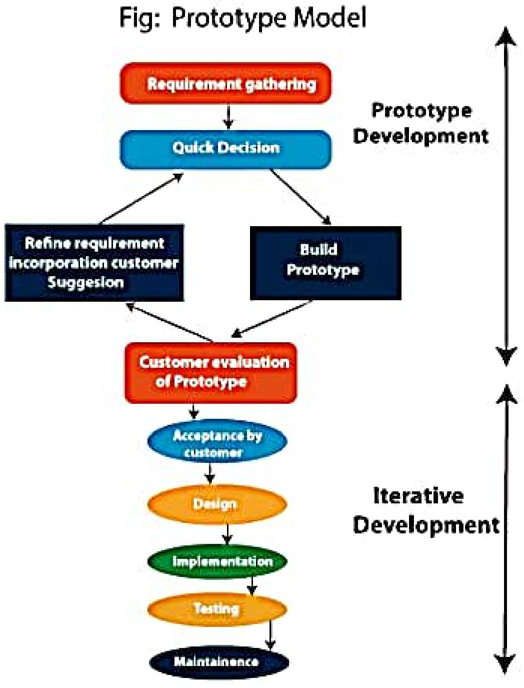
- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Testing and debugging during smaller iteration is easy.
- It is easily acceptable to ever-changing needs of the project.
- Risks are identified and resolved during iteration.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



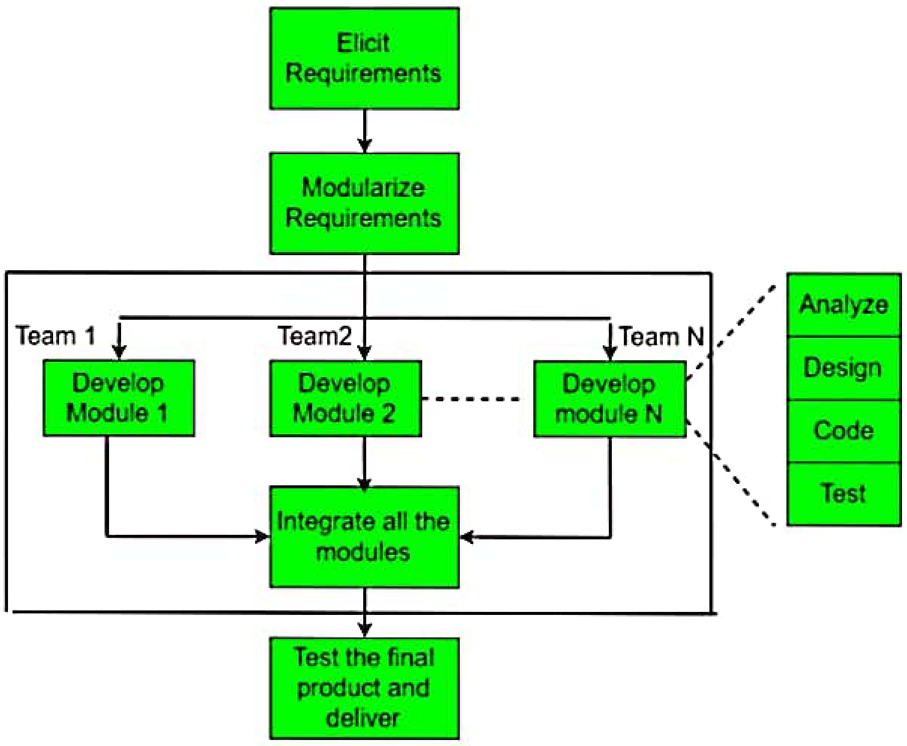
- The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system.
- In many instances, the client only has a general view of what is expected from the software product. In such 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.

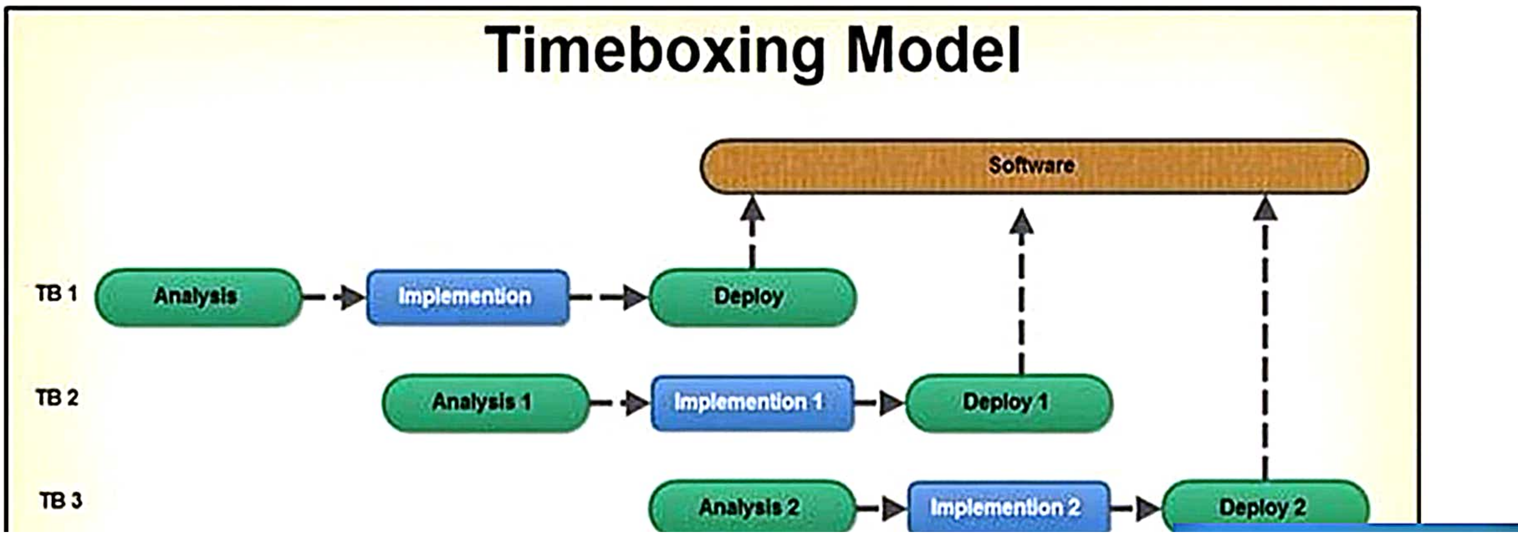
4. **Transition:**
   1. Here finally project is transit from development environment to production.
   2. Here we also set the project on beta testing mode.
   3. Remove the bugs from project based on customer feedback.

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

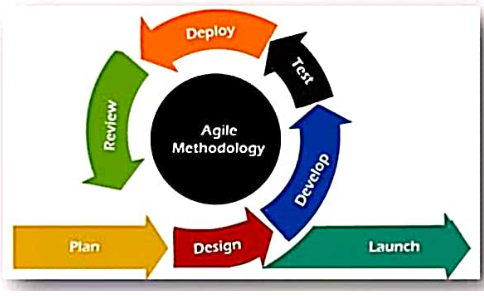
About Agile Model

➢ **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*