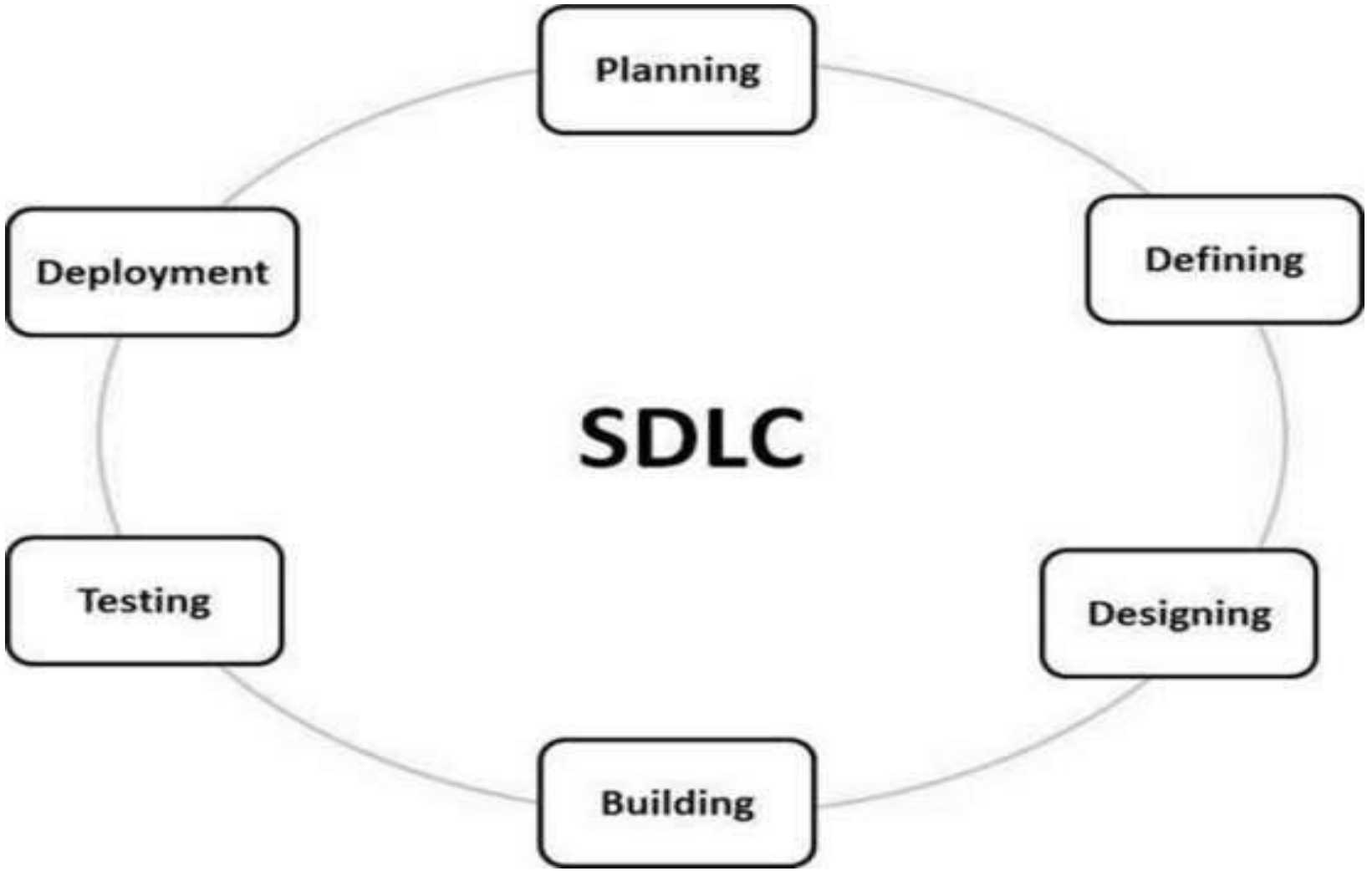
AGILE

SDLC(SOFTWARE DEVELOPMENT LIFE CYCLE)

- Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares.
- The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.
- It is also called as Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.
- SO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

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.


Planning
Defining
Designing
Building
Testing
Deployment
SDLC

PLANNING AND REQIUREMENT 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.
- This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.
- Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage.

Defining Requirements

- Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts.
- This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

Designing the Product Architecture

- SRS is the reference for product architects to come out with the best architecture for the product to be developed.
- Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.
- This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

Building or Developing the Product

- The programming code is generated as per DDS during this stage.
- If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.
- Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code.
- Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding.
- The programming language is chosen with respect to the type of software being developed.

Testing the Product

- The testing activities are mostly involved in all the stages of SDLC.
- product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Deployment in the Market and Maintenance

- Once the product is tested and ready to be deployed it is released formally in the appropriate market.
- Sometimes product deployment happens in stages as per the business strategy of that organization.
- The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).
- After the product is released in the market, its maintenance is done for the existing customer base.

SDLC 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.

- ❑ Waterfall Model
- ❑ Iterative Model
- ❑ Spiral Model
- ❑ V-Model
- ❑ Big Bang Model

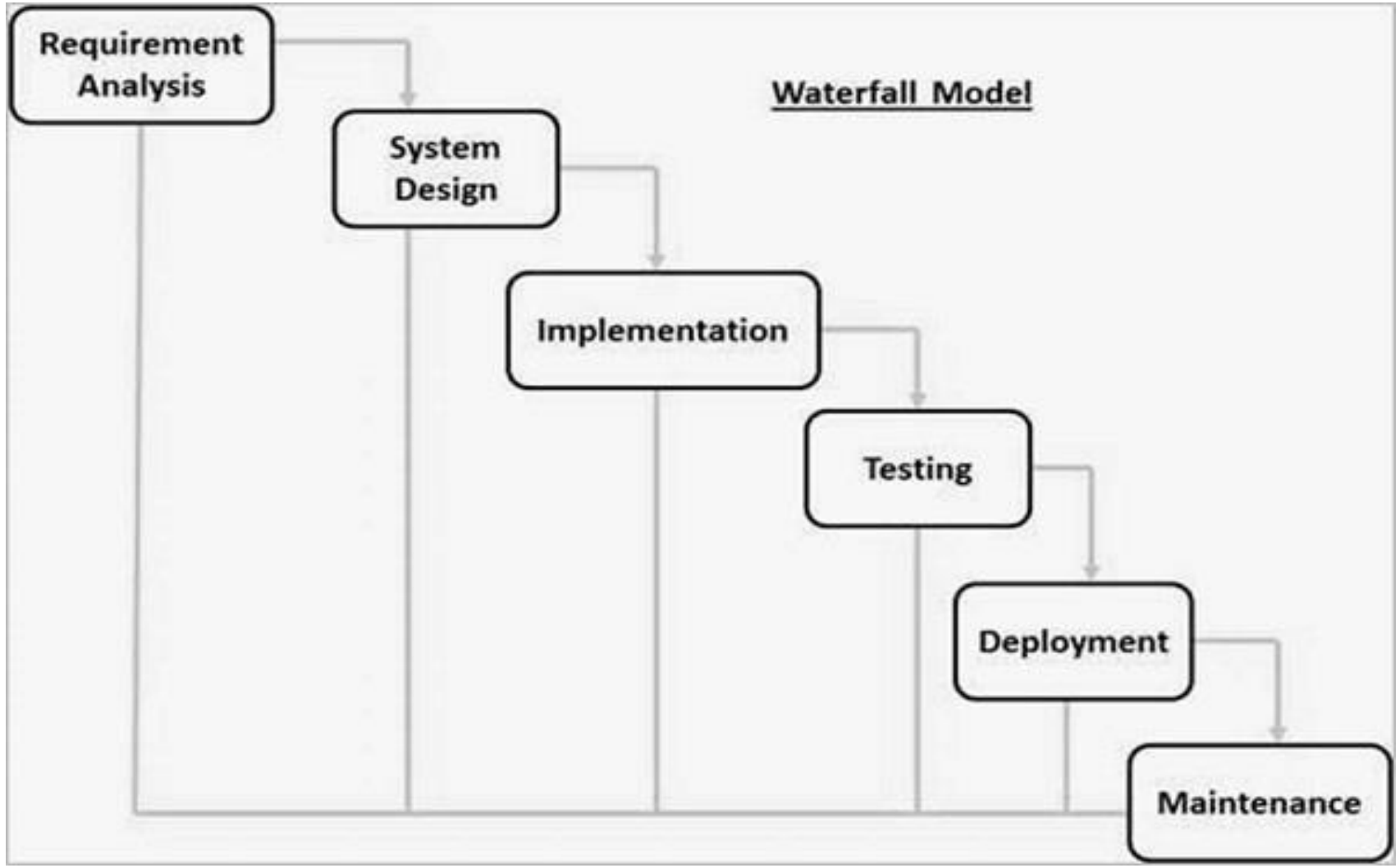
WATER FALL MODEL

- he 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.
- The phases do not overlap.

Waterfall Model - Design

- Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project.
- In "The Waterfall" approach, the whole process of software development is divided into separate phases.
- In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.


Waterfall Model
Requirement Analysis
System Design
Implementation
Testing
Deployment
Maintenance

- **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.

Waterfall Model - Application

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

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.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

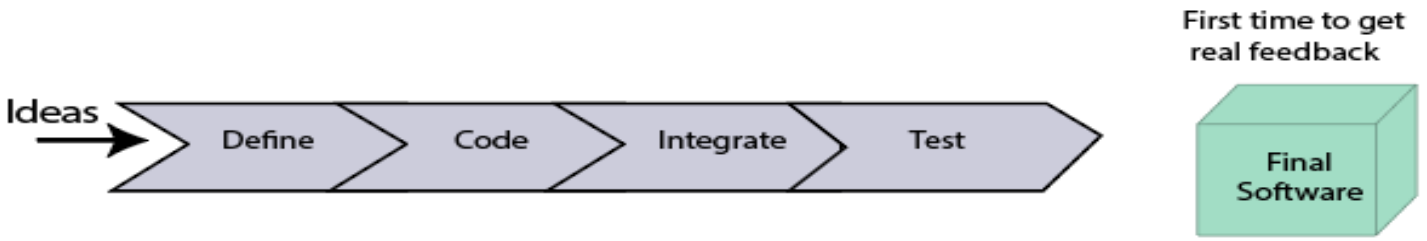
DISADVANTAGES

- 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.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

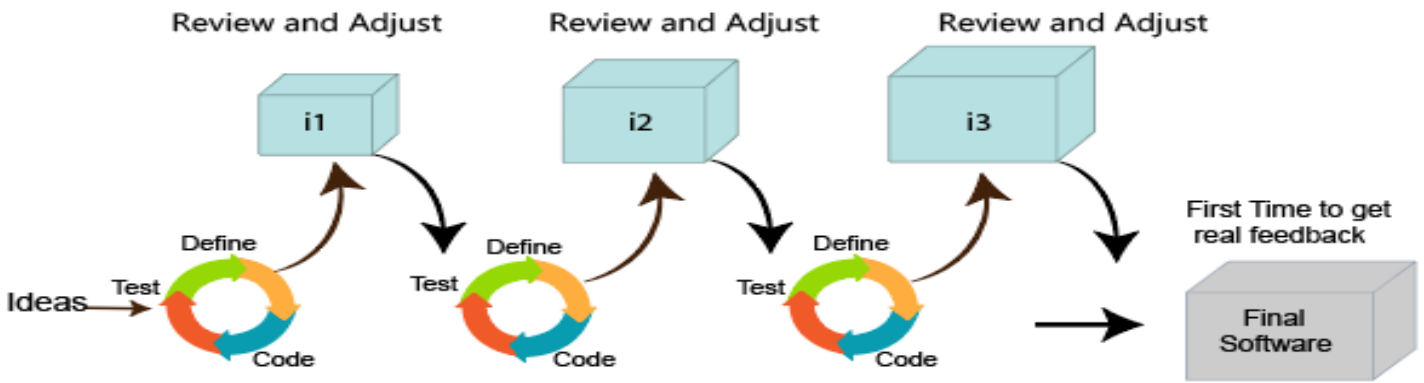
AGILE METHODOLOGY

- An agile methodology is an iterative approach to software development.
- Each iteration of agile methodology takes a short time interval of 1 to 4 weeks.
- The agile development process is aligned to deliver the changing business requirement.
- It distributes the software with faster and fewer changes.
- The single-phase software development takes 6 to 18 months.
- In single-phase development, all the requirement gathering and risks management factors are predicted initially.

AGILE METHODOLOGY



Traditional Method



Agile Method

AGILE DEVELOPMENT MODEL PHASES

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

- **Requirements gathering:**

In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

- **Design the requirements:**

When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system

- **Construction/ iteration:**

When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

- **Testing:**

In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

- **Deployment:**

In this phase, the team issues a product for the user's work environment.

- **Feedback:**

After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

Agile Testing Methods:

- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development
- Extreme Programming(XP)

Agile manifesto

- In February 2001, at the Snowbird resort in Utah, a team of 17 software developers met to discuss lightweight development methods. The result of their meeting was the following Agile Manifesto for software development

✓Individuals and interactions over Processes and tools.

✓Working software over comprehensive documentation.

✓Customers are collaboration over contact negotiation.

✓Responding to change over following a plan.

Principles of agile

- **Customer Satisfaction:**

  Manifesto provides high priority to satisfy the costumer's requirements. This is done through early and continuous delivery of valuable software.

- **Welcome Change:**

  Making changes during software development is common and inevitable. Every changingrequirement should be welcome, evenin the late development phase. Agile process works to increase the customers' competitive advantage

- **Deliver the Working Software:**

Deliver the working software frequently, ranging from a few weeks to a few months with considering the shortest time period.

- **Collaboration:**

Business people (Scrum Master and Project Owner) and developers must work together during the entire life of a project development phase.

- **Motivation:**

Projects should be build around motivated team members. Provide such environment that supportsindividual team members and trust them. It makes them feel responsible for gettingthe job donethoroughly.

- **Face-to-face Conversation:**

Face-to-face conversation betweenScrum Master anddevelopment team and between the Scrum Master and customers for the most efficient and effective method of conveying information to and within a development team.

- **Measure the Progress as per the Working Software:**

The working software is the key and primary measure of the progress.

- **Maintain Constant Pace:**

The aim of agile development is sustainable development. All the businesses and users should be able to maintain a constant pace with the project.

- **Monitoring:**

  Pay regular attention to technical excellence and good design to maximize agility.

- **Simplicity:**

  Keep things simple and use simple terms to measure the work that is not completed.

- **Self-organized Teams:**

  The Agile team should be self-organized. They should not be depending heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams

- **Review the Work Regularly:**

The work should be reviewed at regular intervals, so that the team can reflect on how to become more productive and adjust its behavior accordingly.