SOFTWARE ENGINEERING

UNIT-1

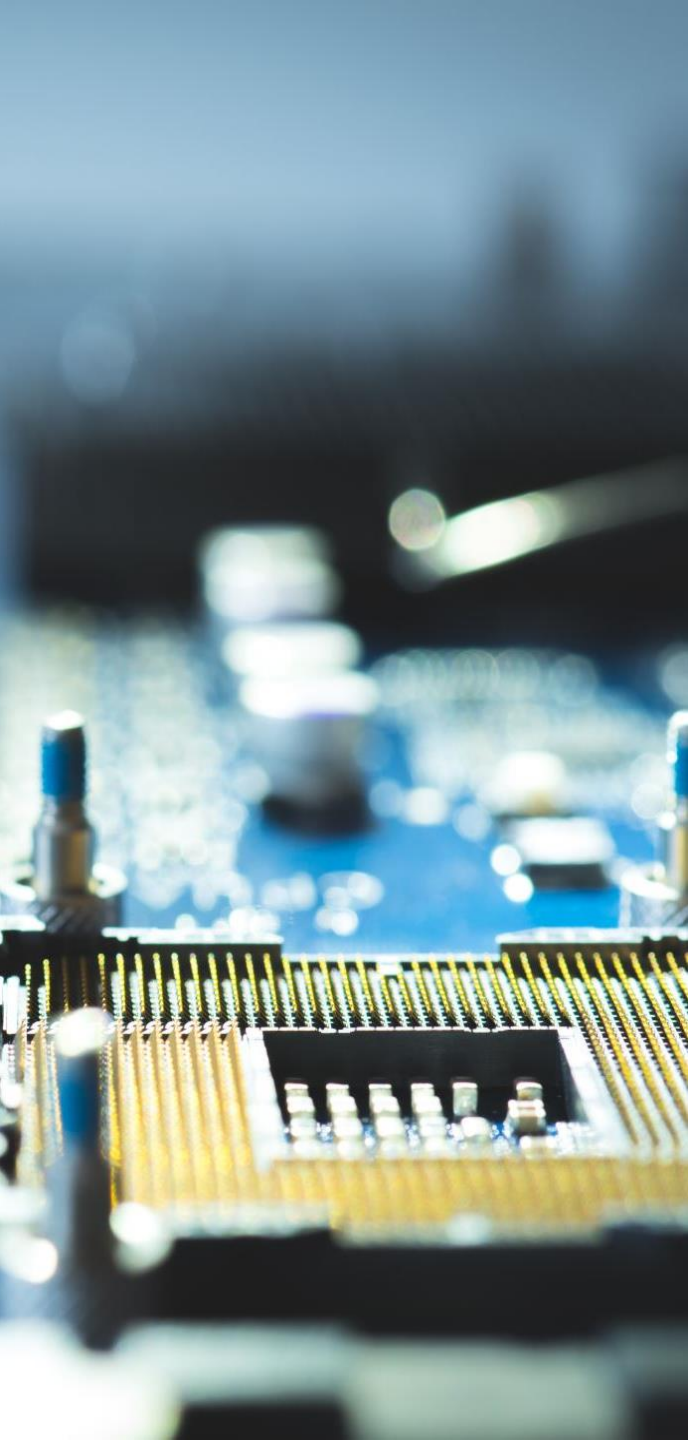
UNIT CONTENT

- **Introduction:** What is software engineering? Software Development Life Cycle, Requirements Analysis, Software Design, Coding, Testing, Maintenance etc.
- **Software Requirements:** Functional and Non-functional requirements, User Requirements, System Requirements, Interface Specification, Documentation of the software requirements.
- **Software Processes:**
- Process and Project, Component Software Processes.
- **Software Development Process Models.**
  - Waterfall Model.
  - Prototyping.
  - Iterative Development.
  - Rational Unified Process.
  - The RAD Model
  - Time boxing Model.

- **Agile software development**: Agile methods, Plan-driven and agile development, Extreme programming, Agile project management, Scaling agile methods.

Introduction To Software

A system usually consisting of several separate programs , configuration files which are used to set up these programs, system documentation describing the structure of the system and user documentation



Types Of Software Product

A. <u>Generic Products</u> –

Stand-alone systems that are produced by a development organization and sold on the open market to any customer.

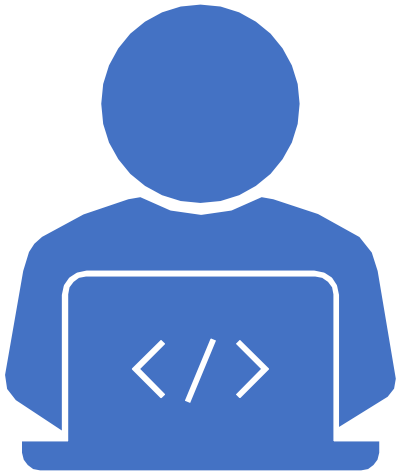
Examples – Databases, Word Processors and Project Management Tools

B. <u>Customized Products</u> –

Systems commissioned by a particular customer. A software contractor develops the software especially for that customer.

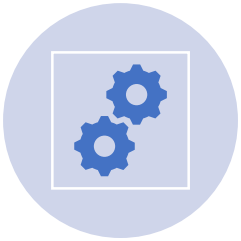
Examples – Control systems for electronic systems and air traffic control systems

What is software engineering



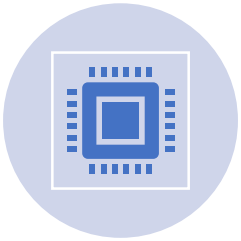
- Software engineering is the process of analysing user needs and designing, constructing, and testing end user applications that will satisfy these needs through the use of software programming languages.
- It is the application of engineering principles to software development.
- In contrast to simple programming, software engineering is used for larger and more complex software systems, which are used as critical systems for businesses and organizations.

Software Development Life Cycle

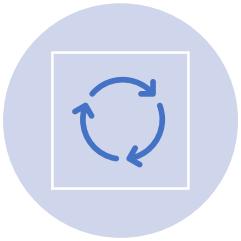


SDLC stands for Software Development Life Cycle.

SDLC is a process followed for a software project, within a software organization.



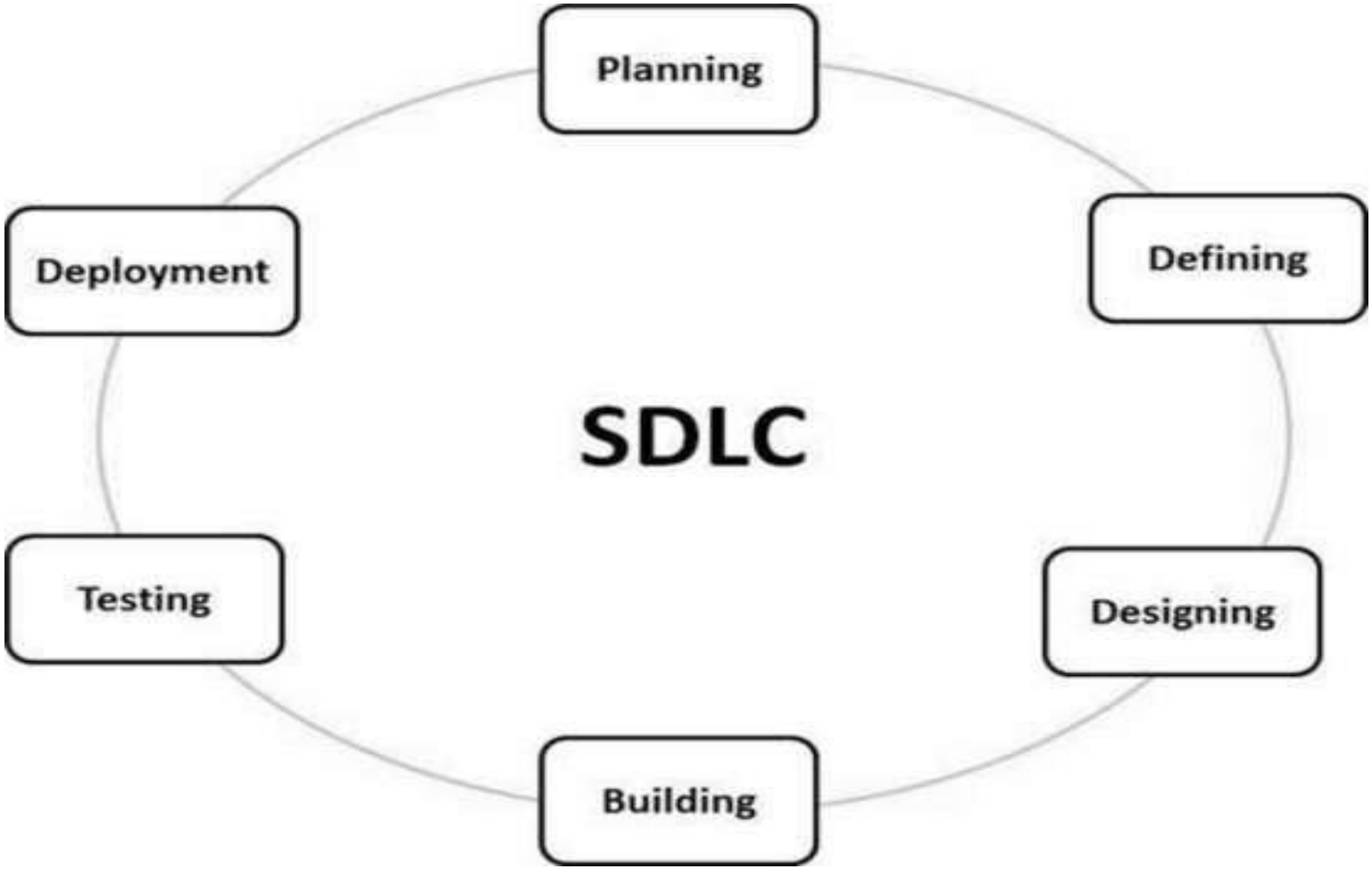
It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software.



The life cycle defines a methodology for improving the quality of software and the overall development process.

Software Development Life Cycle

Various Stages of a typical SDLC.



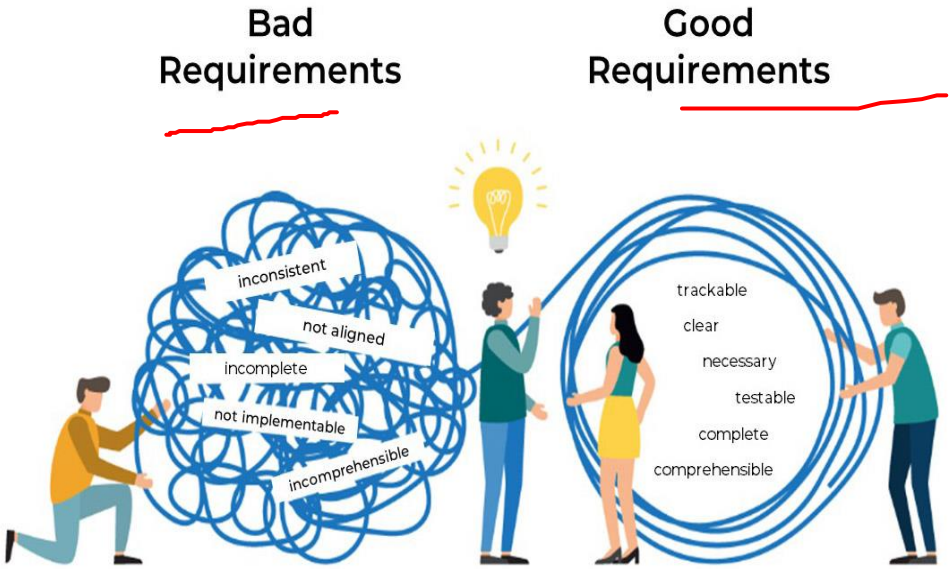
LIFE CYCLE STAGES

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.
- 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.
- The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: 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.



Stage 3: Designing the Product Architecture

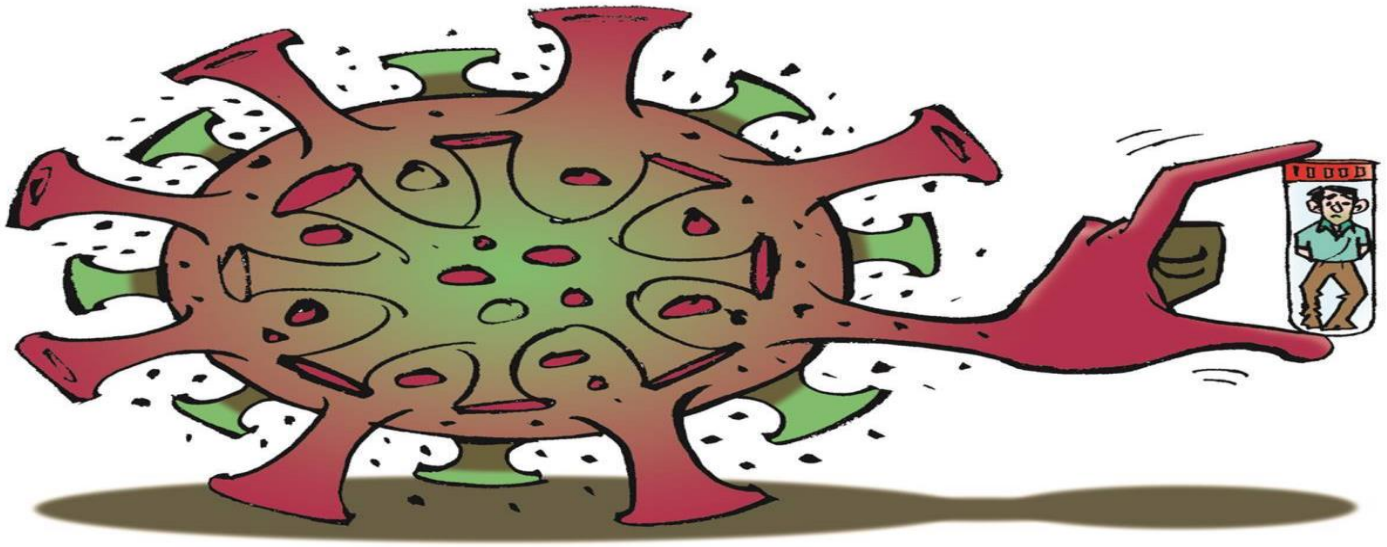
- SRS is the reference for product architects to come out with the best architecture for the product to be developed.
- Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a **DDS - Design Document Specification.**
- A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any).
- The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

- In this stage of SDLC the actual development starts and the product is built.
- 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.

Stage 5: Testing the Product

- This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC.
- 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.**

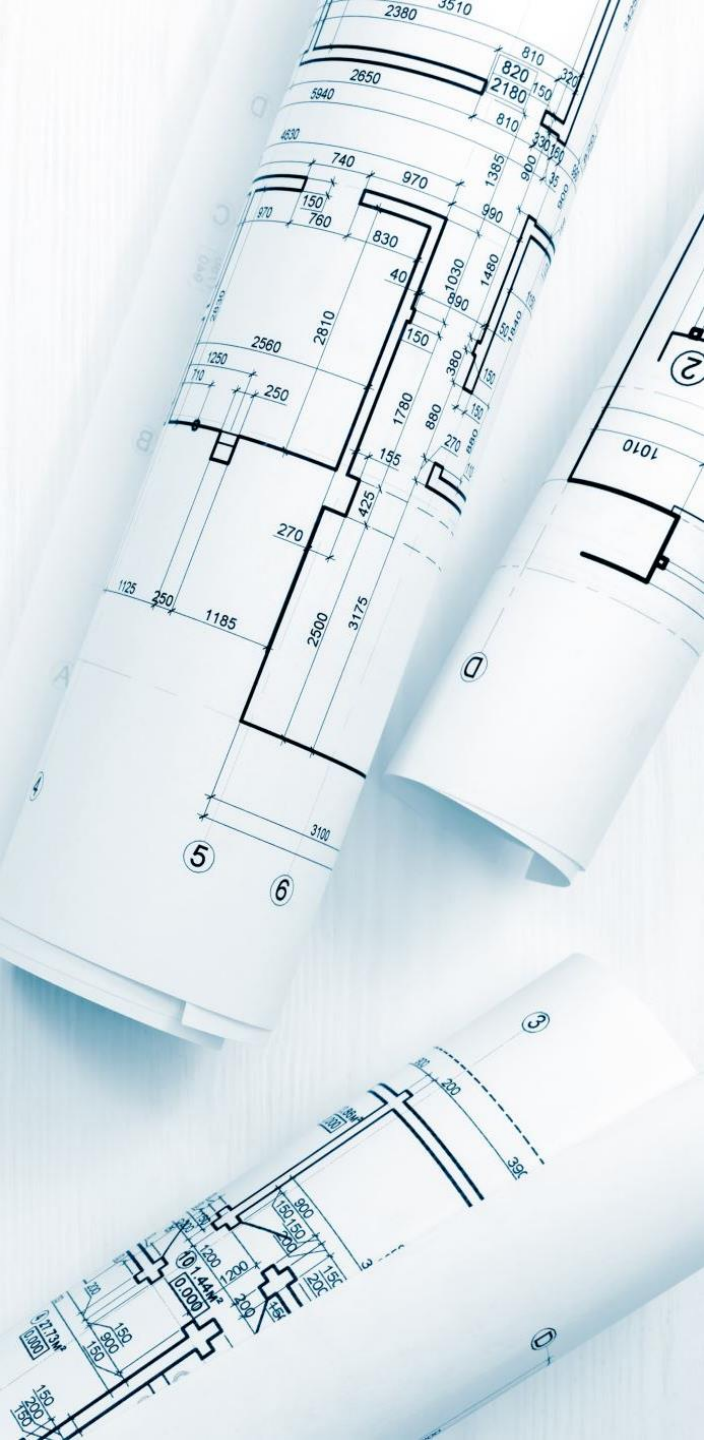


Stage 6: Deployment in the Market and Maintenance

- 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).
- Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment.
- After the product is released in the market, its maintenance is done for the existing customer base.

ESSENTIAL ATTRIBUTES OF GOOD SOFTWARE

Product characteristics	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable, and compatible with other systems that they use.



Requirements Analysis

- **Requirements analysis**, also called **requirements** engineering, is the process of determining user expectations for a new or modified product.
- These features, called **requirements**, must be quantifiable, relevant and detailed.
- In **software** engineering, such **requirements** are often called functional specifications.

Functional Requirement

- A function is described as a set of inputs, the behaviour, and outputs. **Functional requirements** may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish.
- In systems engineering and **requirements** engineering, a **non-functional requirement** (NFR) is a **requirement** that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours.
- They are contrasted with **functional requirements** that define specific behaviour or functions.