# **Module: 1**

# **What is software? What is software engineering?**

Software is a collection of computer programs and related data that provide the instructions for telling a computer what to do and how to do it.

* Software engineering is the art of developing quality software on time and within budget.
* **Software Engineering** is a systematic approach to the design, development, operation, and maintenance of a software system.

# **Explain types of software?**

* System software
* System software provides the basic functions for computer usage and helps to run the computer hardware and system.
* System software is the s/w used by the computer to translate inputs from various sources into a language which a machine can understand.
* Basically, OS coordinates the different hardware components of a computer.
* Ex. Linux, window, macOS, Android, iOS
* **Application s/w:**
* Application software is the general designation of computer programs for performing user tasks. - Types of application s/w
* Mobile app: - Application that run on mobile - Ex. Instagram, Facebook, etc.
* Desktop app: - That run stand-alone in a desktop or laptop computer. - Ex. Microsoft office suite which includes Word, Excel and PowerPoint. - Ex. Outlook for email, and Firefox, Google Chrome, Mozilla are the web browser. - Anti-virus is an application and so is the media player.
* Web app: - That run on a web browser - ex. google.com, facebook.com, etc
* **Programming s/w:**
* Programming software is the process of designing, writing, testing, debugging, and maintaining the source code of computer programs.
* This s/w is pre-written in a programming language.
* The purpose of programming is to create a program that exhibits a certain desired behaviour.
* Ex. c++, html, java, Simlab, php, Python and Visual basic.

1. **What is SDLC? Explain each phase of SDLC**

* SDLC is a structure imposed on the development of a software product that defines the process for planning, implementation, testing, documentation, deployment, and ongoing maintenance and support. There are a number of different development models.
* A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and lifecycle management of an application or piece of software.
* The methodology within the SDLC process can vary across industries and organizations, but standards such as ISO/IEC 12207 represent processes that establish a lifecycle for software, and provide a mode for the development, acquisition, and configuration of software systems.
* **SDLC phase**

|  |  |
| --- | --- |
| Requirements Collection/Gathering | Establish Customer Needs |
| Analysis | Model And specify the requirements- “What” |
| Design | Model And Specify a Solution – “Why” |
| Implementation | Construct a Solution in Software |
| Testing | Validate the solution against the requirements |
| Maintenance | Repair defects and adapt the solution to the new requirements |

* **Requirement Gathering**
* Features
* Usage scenarios
* Although requirements may be documented in written form, they may be incomplete, unambiguous, or even incorrect.
* Requirements will Change!
* Inadequately captured or expressed in the first place
* User and business needs change during the project
* Validation is needed throughout the software lifecycle, not only when the “final system” is delivered.
* Build constant feedback into the project plan
* Plan for change
* Early prototyping [e.g., UI] can help clarify the requirements
* Functional and Non-Functional
* Requirements definitions usually consist of **natural language**, supplemented by (e.g., UML) **diagrams and tables**.
* **Three types of problems can arise:**
* **Lack of clarity:** It is hard to write documents that are both **precise and easy-to-read**.
* **Requirements confusion:Functional and Non-functional** requirements tend to be intertwined.
* **Requirements Amalgamation:** Several **different requirements** may be expressed together.
* **Types of Requirements:**
* **Functional Requirements:** describe system **services or functions**.
* Compute sales tax on a purchase
* Update the database on the server
* **Non-Functional Requirements:** are **constraints** on the system or the development process.
* Non-functional requirements may be more critical than functional requirements.
* If these are not met, the system is useless!
* **Analysis Phase**
* The analysis phase defines the requirements of the system, independent of how these
* Requirements will be accomplished.
* This phase defines the problem that the customer is trying to solve.
* The deliverable result at the end of this phase is a requirement document.
* Ideally, this document states in a clear and precise fashion what is to be built.
* This analysis represents the “**what”** phase.
* The requirement documentaries to capture the requirements from the customer's perspective by defining goals.
* This phase starts with the requirement document delivered by the requirement phase and mapsthe requirements into architecture.
* The architecture defines the components, their interfaces and behaviours.
* The deliverable design document is the architecture.
* This phase represents the “**how”** phase.
* Details on computer programming languages and environments, machines, packages, application architecture, distributed architecture layering, memory size, platform, algorithms, data structures, global type definitions, interfaces, and many other engineering details are established.
* The design may include the usage of existing components.
* **Design Phase**
* Design Architecture Document
* Implementation Plan
* Critical Priority Analysis
* Performance Analysis
* Test Plan
* The Design team can now expand upon the information established in the requirement
* Document.
* The requirement document must guide this decision process.
* Analysing the trade-offs of necessary complexity allows for many things to remain simple which, in turn, will eventually lead to a higher quality product. The architecture team also converts the typical scenarios into a test plan.
* **Implementation Phase**
* In the implementation phase, the team builds the components either from scratch or by composition.
* Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility.
* For example, a component may be narrowly designed for this particular system, or the
* Component may be made more general to satisfy a reusability guideline.
* Implementation – Code
* Critical Error Removal
* The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging.
* The end deliverable is the product itself. There are already many established techniques associated with implementation.

* **Testing Phase**
* Simply stated, quality is very important. Many companies have not learned that quality is important and deliver more claimed functionality but at a lower quality level.
* It is much easier to explain to a customer why there is a missing feature than to explain to a customer why the product lacks quality.
* A customer satisfied with the quality of a product will remain loyal and wait for new Functionality in the next version.
* Quality is a distinguishing attribute of a system indicating the degree of excellence.
* Regression Testing
* Internal Testing
* Unit Testing
* Application Testing
* Stress Testing
* The testing phase is a separate phase which is performed by a different team after the implementation is completed.
* There is merit in this approach; it is hard to see one‘s own mistakes, and a fresh eye can discover obvious errors much faster than the person who has read and re-read the material many times.
* Unfortunately, delegating (alternate) testing to another team leads to as lack (dull) attitude regarding quality by the implementation team.
* If the teams are to be known as craftsmen, then the teams should be responsible for establishing high quality across all phases.
* An attitude change must take place to guarantee quality. Regardless if testing is done after the-fact or continuously, testing is usually based on a regression technique split into several major focuses, namely internal, unit, application, and stress.
* **Maintenance Phase**
* Software maintenance is one of the activities in software engineering, and is the process of enhancing and optimizing deployed software (software release), as well as fixing defects.
* Software maintenance is also one of the phases in the System Development Life Cycle (SDLC), as it applies to software development. The maintenance phase is the phase which comes after deployment of the software into the field.
* The developing organization or team will have some mechanism to document and track Defects and deficiencies.
* configuration and version management
* reengineering (redesigning and refactoring)
* updating all analysis, design and user documentation
* Repeatable, automated tests enable evolution and refactoring

Maintenance is the process of changing a system after it has been deployed.

* **Corrective maintenance: identifying and repairing defects**
* **Adaptive maintenance:** adapting the existing solution to the **new platforms**.
* **Perfective Maintenance:** implementing the **new requirements**

In a spiral lifecycle, everything after the delivery and deployment of the first prototype can be considered “maintenance”!

1. **What is DFD? Create a DFD diagram on Flipkart**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

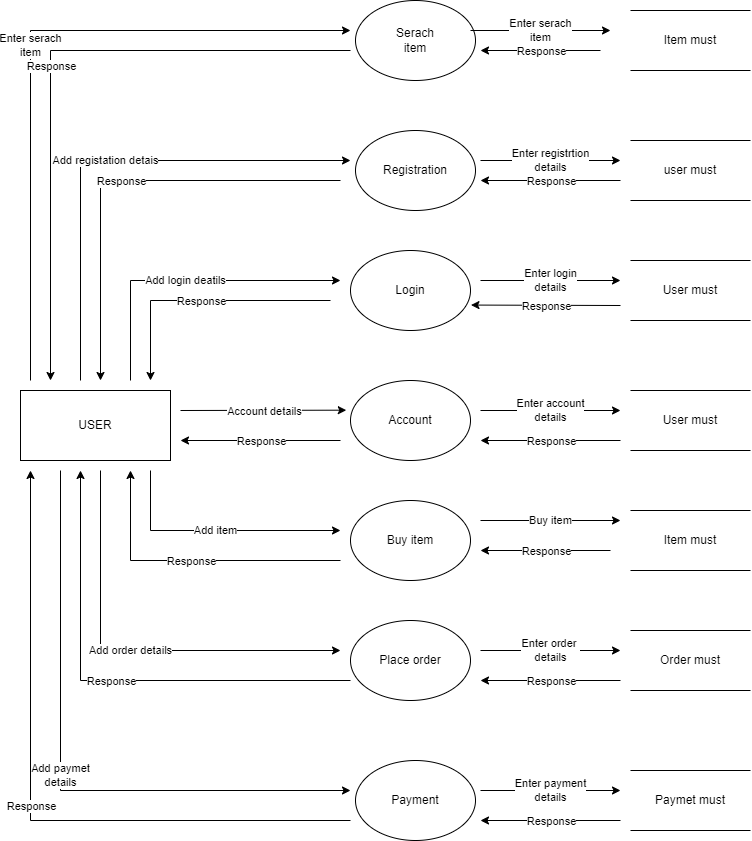


Fig 1 [DFD diagram on flipkart]

1. **What is Flow chart? Create a flowchart to make addition of two numbers**

A flow chart is a graphical or symbolic representation of a process. Each step in the process is represented by a different symbol and contains a short description of the process step. The flow chart symbols are linked together with arrows showing the process flow direction.

A flowchart is a graphical representation of the operations involved in a data processing system.

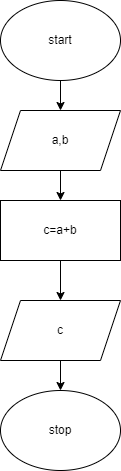


Fig 2 [flowchart of addition of two number]

1. **What is Use case Diagram? Create a use-case on bill payment on paytm.**

A **use case diagram** is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

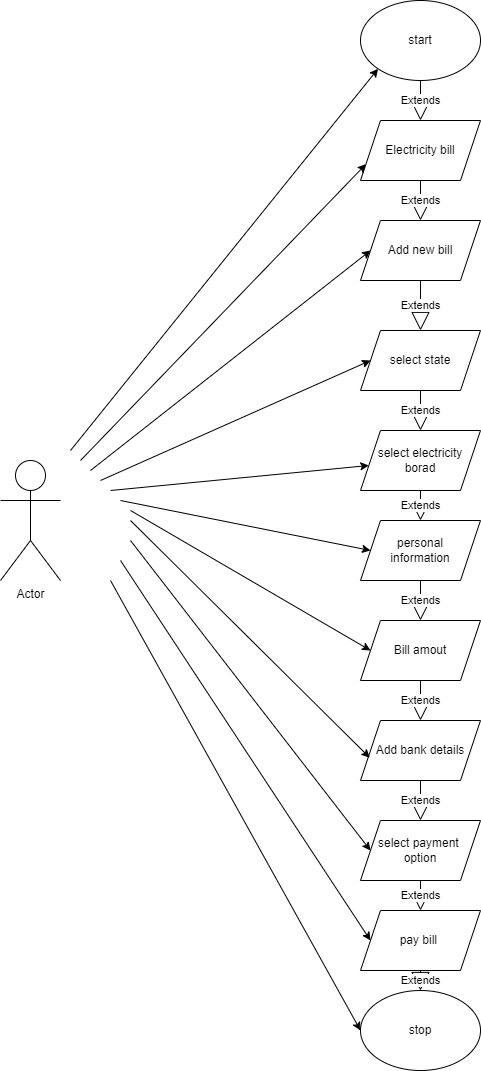


Fig 3 [use case diagram of bill payment on paytm]