1. What is software engineering, and how does it differ from traditional programming? Software Development Life Cycle (SDLC):

Software engineering is the branch of computer science that deals with the design, development, testing and maintenance of software applications. Whereas software engineering consists of all aspects of software creation, including concept design and coding

1. Explain the various phases of the Software Development Life Cycle. Provide a brief description of each phase. Agile vs. Waterfall Models:

**Stage 1: Project Planning**

The first stage of SDLC is all about “What do we want?” Project planning is a vital role in the software delivery lifecycle since this is the part where the team estimates the cost and defines the requirements of the new software.

**Stage 2: Gathering Requirements & Analysis**

The second step of SDLC is gathering maximum information from the client requirements for the product. Discuss each detail and specification of the product with the customer. The development team will then analyse the requirements keeping the design and code of the software in mind.

**Stage 3: Design**

In the design phase (3rd step of SDLC), the program developer scrutinizes whether the prepared software suffices all the requirements of the end-user. Additionally, if the project is feasible for the customer technologically, practically, and financially.

**Stage 4: Coding or Implementation**

Time to code! It means translating the design to a computer-legible language. In this fourth stage of SDLC, the tasks are divided into modules or units and assigned to various developers. The developers will then start building the entire system by writing code using the programming languages they chose.

**Stage 5: Testing**

Once the developers build the software, then it is deployed in the testing environment. Then the testing team tests the functionality of the entire system. In this fifth phase of SDLC, the testing is done to ensure that the entire application works according to the customer requirements.

**Stage 6: Deployment**

The sixth phase of SDLC: Once the testing is done, and the product is [ready for deployment](https://www.betsol.com/blog/how-to-make-software-deployments-easier/), it is released for customers to use. The size of the project determines the complexity of the deployment. The users are then provided with the training or documentation that will help them to operate the software.

**Stage 7: Maintenance**

The actual problem starts when the customer actually starts using the developed system and those needs to be solved from time to time. Maintenance is the seventh phase of SDLC where the developed product is taken care of. According to the changing user end environment or technology, the software is updated timely.

1. **Waterfall Model:**This SDLC model is considered to be the oldest and most forthright. We finish with one phase and then start with the next, with the help of this methodology. Why the name waterfall? Because each of the phases in this model has its own mini-plan and each stage waterfalls into the next. A drawback that holds back this model is that even the small details left incomplete can hold an entire process.

**Agile Model**: Agile is the new normal; It is one of the most utilized models, as it approaches software development in incremental but rapid cycles, commonly referred to as “sprints”. With new changes in scope and direction being implemented in each sprint, the project can be completed quickly with higher flexibility. Agile means spending less time in the planning phases, and a project can diverge from original specifications.

1. **Requirements Engineering** is the process of identifying, eliciting, analysing, specifying, validating, and managing the needs and expectations of stakeholders for a software system.

**Feasibility Study**

The feasibility study mainly concentrates on below five mentioned areas below. Among these Economic Feasibility Study is the most important part of the feasibility analysis and the Legal Feasibility Study is less considered feasibility analysis.

1. **Technical Feasibility**: In Technical Feasibility current resources both hardware software along required technology are analyzed/assessed to develop the project. This technical feasibility study reports whether there are correct required resources and technologies that will be used for project development. Along with this, the feasibility study also analyzes the technical skills and capabilities of the technical team, whether existing technology can be used or not, whether maintenance and up-gradation are easy or not for the chosen technology, etc.
2. **Operational Feasibility:**In Operational Feasibility degree of providing service to requirements is analyzed along with how easy the product will be to operate and maintain after deployment. Along with this other operational scopes are determining the usability of the product, Determining suggested solution by the software development team is acceptable or not, etc.
3. **Economic Feasibility:**In the Economic Feasibility study cost and benefit of the project are analyzed. This means under this feasibility study a detailed analysis is carried out will be cost of the project for development which includes all required costs for final development hardware and software resources required, design and development costs operational costs, and so on. After that, it is analyzed whether the project will be beneficial in terms of finance for the organization or not.

**Requirements Elicitation**

It is related to the various ways used to gain knowledge about the project domain and requirements. The various sources of domain knowledge include customers, business manuals, the existing software of the same type, standards, and other stakeholders of the project. The techniques used for requirements elicitation include interviews, brainstorming, task analysis, Delphi technique, prototyping, etc. Some of these are discussed [here.](https://www.geeksforgeeks.org/software-engineering-requirements-elicitation/) Elicitation does not produce formal models of the requirements understood. Instead, it widens the domain knowledge of the analyst and thus helps in providing input to the next stage.

Requirements elicitation is the process of gathering information about the needs and expectations of stakeholders for a software system. This is the first step in the requirements engineering process and it is critical to the success of the software development project. The goal of this step is to understand the problem that the software system is intended to solve and the needs and expectations of the stakeholders who will use the system.

Several techniques can be used to elicit requirements, including:

* **Interviews**: These are one-on-one conversations with stakeholders to gather information about their needs and expectations.
* **Surveys**: These are questionnaires that are distributed to stakeholders to gather information about their needs and expectations.
* **Focus Groups**: These are small groups of stakeholders who are brought together to discuss their needs and expectations for the software system.
* **Observation**: This technique involves observing the stakeholders in their work environment to gather information about their needs and expectations.
* **Prototyping**: This technique involves creating a working model of the software system, which can be used to gather feedback from stakeholders and to validate requirements.

It’s important to document, organize, and prioritize the requirements obtained from all these techniques to ensure that they are complete, consistent, and accurate.

**Requirements Specification**

This activity is used to produce formal software requirement models. All the requirements including the functional as well as the non-functional requirements and the constraints are specified by these models in totality. During specification, more knowledge about the problem may be required which can again trigger the elicitation process. The models used at this stage include ER diagrams, data flow diagrams(DFDs), function decomposition diagrams(FDDs), data dictionaries, etc.

Requirements specification is the process of documenting the requirements identified in the analysis step in a clear, consistent, and unambiguous manner. This step also involves prioritizing and grouping the requirements into manageable chunks.

The goal of this step is to create a clear and comprehensive document that describes the requirements for the software system. This document should be understandable by both the development team and the stakeholders.

**Several types of requirements are commonly specified in this step, including**

1. [**Functional Requirements**](https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/)**:** These describe what the software system should do. They specify the functionality that the system must provide, such as input validation, data storage, and user interface.
2. [**Non-Functional Requirements**:](https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/) These describe how well the software system should do it. They specify the quality attributes of the system, such as performance, reliability, usability, and security.
3. **Constraints:**These describe any limitations or restrictions that must be considered when developing the software system.
4. **Acceptance Criteria**: These describe the conditions that must be met for the software system to be considered complete and ready for release.

To make the requirements specification clear, the requirements should be written in a natural language and use simple terms, avoiding technical jargon, and using a consistent format throughout the document. It is also important to use diagrams, models, and other visual aids to help communicate the requirements effectively.

Once the requirements are specified, they must be reviewed and validated by the stakeholders and development team to ensure that they are complete, consistent, and accurate.

**Requirements Verification and Validation**

**Verification:** It refers to the set of tasks that ensures that the software correctly implements a specific function.

**Validation:** It refers to a different set of tasks that ensures that the software that has been built is traceable to customer requirements. If requirements are not validated, errors in the requirement definitions would propagate to the successive stages resulting in a lot of modification and rework. The main steps for this process include:

1. The requirements should be consistent with all the other requirements i.e. no two requirements should conflict with each other.
2. The requirements should be complete in every sense.
3. The requirements should be practically achievable.

Reviews, buddy checks, making test cases, etc. are some of the methods used for this.

Requirements verification and validation (V&V) is the process of checking that the requirements for a software system are complete, consistent, and accurate and that they meet the needs and expectations of the stakeholders. The goal of V&V is to ensure that the software system being developed meets the requirements and that it is developed on time, within budget, and to the required quality.

1. Verification is checking that the requirements are complete, consistent, and accurate. It involves reviewing the requirements to ensure that they are clear, testable, and free of errors and inconsistencies. This can include reviewing the requirements document, models, and diagrams, and holding meetings and walkthroughs with stakeholders.
2. Validation is the process of checking that the requirements meet the needs and expectations of the stakeholders. It involves testing the requirements to ensure that they are valid, and that the software system being developed will meet the needs of the stakeholders. This can include testing the software system through simulation, testing with prototypes, and testing with the final version of the software.
3. Verification and Validation is an iterative process that occurs throughout the software development life cycle. It is important to involve stakeholders and the development team in the V&V process to ensure that the requirements are thoroughly reviewed and tested.

It’s important to note that V&V is not a one-time process, but it should be integrated and continue throughout the software development process and even in the maintenance stage.

**Requirements Management**

Requirement management is the process of analysing, documenting, tracking, prioritizing, and agreeing on the requirement and controlling the communication with relevant stakeholders. This stage takes care of the changing nature of requirements. It should be ensured that the SRS is as modifiable as possible to incorporate changes in requirements specified by the end users at later stages too. Modifying the software as per requirements in a systematic and controlled manner is an extremely important part of the requirements engineering process.

Requirements management is the process of managing the requirements throughout the software development life cycle, including tracking and controlling changes, and ensuring that the requirements are still valid and relevant. The goal of requirements management is to ensure that the software system being developed meets the needs and expectations of the stakeholders and that it is developed on time, within budget, and to the required quality.

Several key activities are involved in requirements management, including:

1. **Tracking and controlling changes:** This involves monitoring and controlling changes to the requirements throughout the development process, including identifying the source of the change, assessing the impact of the change, and approving or rejecting the change.
2. **Version control**: This involves keeping track of different versions of the requirements document and other related artifacts.
3. **Traceability**: This involves linking the requirements to other elements of the development process, such as design, testing, and validation.
4. **Communication:**This involves ensuring that the requirements are communicated effectively to all stakeholders and that any changes or issues are addressed promptly.
5. **Monitoring and reporting**: This involves monitoring the progress of the development process and reporting on the status of the requirements.

5. Modularity is a fundamental design principle in software engineering aimed at creating software in a way that minimizes dependencies among the components of a system. This helps to localize the impact of changes, simplifies maintenance, and enhances the understandability of the system.

1. 6. [**Unit Testing**](https://www.geeksforgeeks.org/unit-testing-software-testing/)**:** In this type of testing, errors are detected individually from every component or unit by individually testing the components or units of software to ensure that they are fit for use by the developers. It is the smallest testable part of the software.
2. [**Integration Testing**](https://www.geeksforgeeks.org/software-engineering-integration-testing/)**:** In this testing, two or more modules which are unit tested are integrated to test i.e., technique interacting components, and are then verified if these integrated modules work as per the expectation or not, and interface errors are also detected.
3. [**System Testing**](https://www.geeksforgeeks.org/system-testing/)**:** In system testing, complete and integrated Softwares are tested i.e., all the system elements forming the system are tested as a whole to meet the requirements of the system.
4. [**Acceptance Testing**](https://www.geeksforgeeks.org/acceptance-testing-software-testing/)**:** This is a kind of testing conducted to ensure that the requirements of the users are fulfilled before its delivery and that the software works correctly in the user’s working environment.

After all that information, we can really get to why software testing really is important. Though you might already have an idea from the points we’ve made above, we’ll make a quick list of reasons as to why you need software testing.

7. Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.

As we know that a software product is developed in collaboration by a group of developers they might be located at different locations and each one of them contributes to some specific kind of functionality/features. So in order to contribute to the product, they made modifications to the source code(either by adding or removing). A version control system is a kind of software that helps the developer team to efficiently communicate and manage(track) all the changes that have been made to the source code along with the information like who made and what changes have been made.

**Github**

GitHub is a version control system and repository to host Git projects. If you’re wondering what Git is, it is an open-source version control system authored by Linus Torvalds. Yes, the man behind Linux.

Github is known as the “social media” for software developers. But above all, it helps software teams collaborate and maintain the entire history of code changes.

Strengths:

* Provides code review that is available in a private cloud with GitHub Enterprise or on-premise deployment
* Uses GitHub Marketplace to look for integrations and tools needed to improve your development process further
* Offers a secured environment through signed commits, required status checks, and protected branches to maintain high-standard coding practices
* Enhances collaboration through code review tools where members can discuss implementation details before applying code changes
* Provides high visibility for all the code parts that are new, edited, or deleted. These are all highlighted to spot changes easily and compare versions of code side by side.

**Gitlab**

If your project requires continuous integration and continuous deployment (CI/CD), then GitLab is for you. It is a version control software built for the DevOps lifecycle. Meaning, your projects are quite complex, and it involves cloud engineering and the like.

Gitlab provides basic features such as view code, pull requests, and merged resolution.

Strengths:

* Improves productivity by cutting down the DevOps cycle time, reducing manual work, and connecting silos and stages
* Provides audit management, cycle analytics, issue board and trackers, source code management, [unit testing](https://fullscale.io/blog/unit-testing/), and continuous delivery
* Lowers development risks through frequent deployments that will lead to better predictability, easier troubleshooting, and better code quality
* Defies geographical barriers as distributed, and remote teams can work on the latest version of the code and stay current.

**Beanstalk**

For projects that involve outsourcing, Beanstalk is an ideal option. This [version control software](https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control) uses browser and cloud. It allows users to code, commit, review, and deploy using a browser. For ease of use, you can integrate it with any messaging or email platform for efficient collaborations.

It is an enterprise-class secured infrastructure, with high performance, and reliability. This is ideal for startups businesses.

Strengths:

* Provides robust encryption, two-factor authentication, and password protection functionalities
* Caters to any team size, and can provide a [repository](https://fullscale.io/blog/software-repository/) and branch-level permissions for teams and individuals
* Allows team members are well-informed throughout the deployment because of its release notes feature.

**HelixCore**

HelixCore offers seamless team collaboration and support for both centralized and distributed development workflows. It is available on both cloud and on-premise deployments, making it scalable.

Strengths:

* Provides workflow freedom where members can work remotely but have total control by setting granular file-level permissions
* Supports multi-factor authentication and other security features to secure your intellectual property
* Offers a DevOps-ready environment for continuous integration and scalability.

**AWS CodeCommit**

If you are building a project using AWS, then AWS CodeCommit is for you. It can host secure and scalable private Git repositories. You can seamlessly connect with other products from Amazon Web Services (AWS) and host the code in secured AWS environments.

Strengths:

* Offers efficient [code collaboration](https://fullscale.io/blog/version-control-and-code-collab/) with its full support for pull requests; its users can easily comment and review code changes before merging them into branches
* Provides robust integrations with Identity Access Management(IAM), other AWS services, and third-party services
* Offers a secure repository for your project documents and source code in the cloud.

**Microsoft Team Foundation Server**

Team Foundation Server is an enterprise-grade tool to help project development. It manages source code, tracks and monitors workflows, and ships software. It works with known programming languages and helps accelerate production through its additional features (Team Build, Team Project Portal, Team Foundation Shared Services, etc.).

Strengths:

* Offers a flexible and distributed version control system
* Provides a secured and centralized version control for workflows and code management
* Enables quick reviewing and merging of code in a Git team project through the pull request method.

**Mercurial**

Mercurial is a free and distributed version control software known for its efficiency. It can cater to projects of different sizes. And developers love its simple and intuitive user interface

Strengths:

* Provides a robust backup system, search functionality, project tracking, and management
* Offers data import, export, and migration tools
* Features robust workflow management, including history tracking, security management, and more.

**Concurrent Versions System**

Commercial and open-source developers use CVS. Why? It is one of the oldest version control systems. It is a sophisticated version control software and continues to improve its capabilities.

Strengths:

* Maps out components and modules in larger software projects
* Caters to multiple developers working simultaneously on a project
* Enforces site-specific policies to provide secured operations.

**Bitbucket**

If you are working with projects using Atlassian tools, consider using Bitbucket. It offers code branching, in-line commenting and discussions, and pull requests.

Strengths:

* Works with a local server, data center, and on the cloud
* Integrates with other Atlassian products such as Jira; it lets you create branches directly from the Jira board
* Offers built-in continuous integration and continuous delivery.

Works with a local server, data center, and on the cloud

A software project manager is the most important person inside a team who takes the overall responsibilities to manage the software projects and plays an important role in the successful completion of the projects. This article focuses on discussing the role and responsibilities of a software project manager.

Software Maintenance refers to the process of modifying and updating a software system after it has been delivered to the customer. It is a critical part of the [software development life cycle (SDLC)](https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/) and is necessary to ensure that the software continues to meet the needs of the users over time. This article focuses on discussing Software Maintenance in detail.