**Name: FYNEBOY NWACHUKWU MACDICKSON CELT**

**Cohort: MAY**

**# SE-Assignment-2**

**Assignment: Introduction to Software Engineering**

**Questions:**

**Define Software Engineering:**

**Answer:**

This is the engineering approach toward developing software that can solve problem.

e.g MatLab,Excel etc

**What is software engineering, and how does it differ from traditional programming?**

**Software Development Life Cycle (SDLC):**

Answer:

Software engineering is the engineering development process of software.

e.g The software development process involves several stages

1. Identifying requirements
2. Planning and designing the structure and features of the software
3. Writing and coding the software
4. Testing the software to ensure that it works correctly and meets the intended requirements
5. Deploying the software for use

In traditional programming, a programmer manually provides specific instructions to the computer based on their understanding and analysis of the problem. If the data or the problem changes, the programmer needs to manually update the code.

To differentiate, software engineering envelops all the processes that is required to produce a successful software while TP is the coding process of inputting variables that are required for making it user friend which entails styling, API creation and its integrations.

**Explain the various phases of the Software Development Life Cycle. Provide a brief description of each phase.**

**ANSWER:**

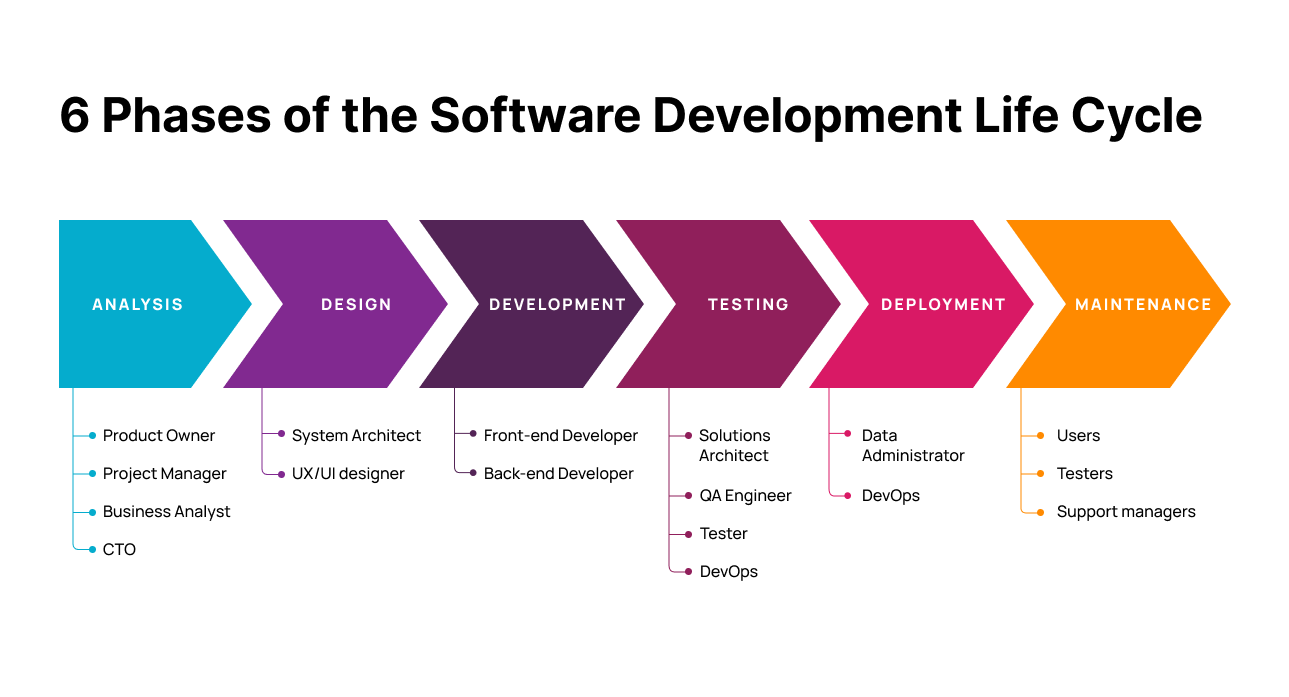


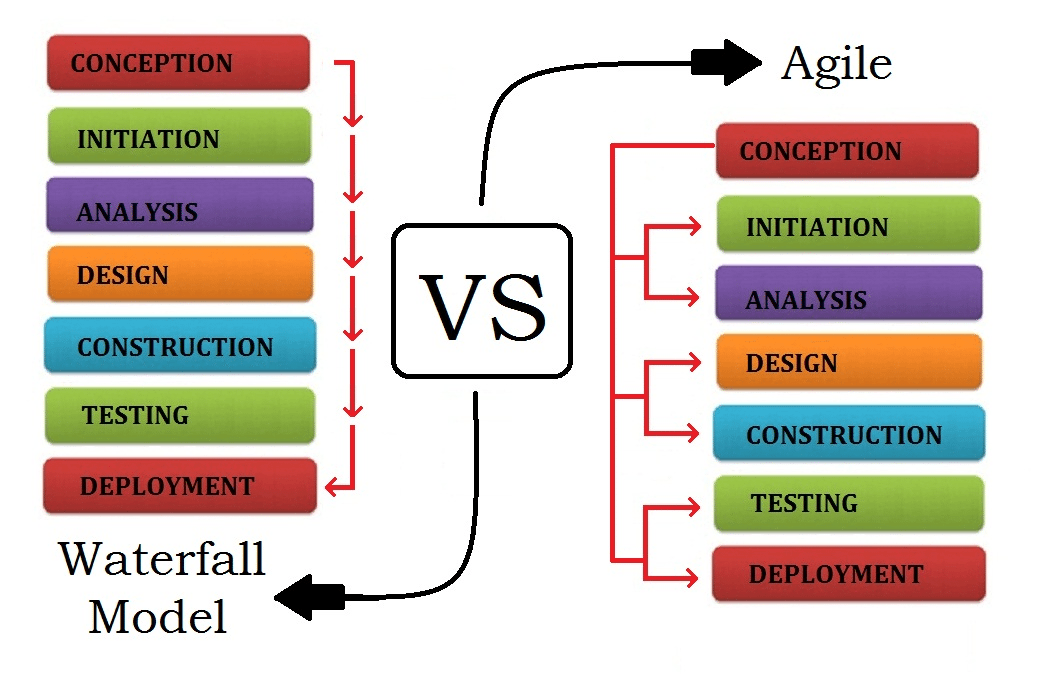
Figure 1https://cdn.brocoders.com/6\_phases\_of\_software\_development\_life\_cycle\_ce25a52c62.png

**Agile vs. Waterfall Models:**

**ANSWER:**

Agile and Waterfall models have different approaches to project management

* Agile is incremental and iterative, focusing on adapting to change and delivering work in iterations and increments.
* Waterfall is linear and sequential, with all planning and requirements gathering done at the beginning of the project.



**Compare and contrast the Agile and Waterfall models of software development. What are the key differences, and in what scenarios might each be preferred?**

**ANSWER**

Certainly! The key difference between Agile and Waterfall lies in their approach to project management:

Waterfall:

1. Linear Process: Waterfall follows a sequential path with well-defined stages. Each stage must be completed before moving to the next.
2. Requirements First: All requirements are gathered upfront at the project’s beginning.
3. Formal Hand-offs: There are formal hand-offs between stages.
4. Final Release: The project lasts until completion, culminating in a final release.

Agile:

1. Iterative and Flexible: Agile focuses on rapid iteration, autonomy, and flexibility.
2. Sprints: Work is divided into time-based bursts called Sprints (typically 1-4 weeks). Progress is made during each Sprint.
3. Self-Organizing Teams: Agile teams self-organize, consulting with stakeholders as needed.
4. Adapting to Change: Agile adapts to change and delivers work incrementally.

**Requirements Engineering:**

Requirements engineering (RE) is the art of defining and designing software systems based on user needs and goals

**What is requirements engineering? Describe the process and its importance in the software development lifecycle.**

Requirement engineering is a process that is performed in the initial stages of any software development. It includes analyzing the customer’s requirement and various tasks such as:

* At first, identify the user’s requirement, what does it want the operational system to perform?
* Analyze the requirements gathered from the customer.
* Evaluate the feasibility of the operational system.
* Propose some unambiguous solutions.
* Validate the requirement provided by the customer.
* Manage the requirements as far as they are modelled into an operational system.

Importance of Requirement Engineering

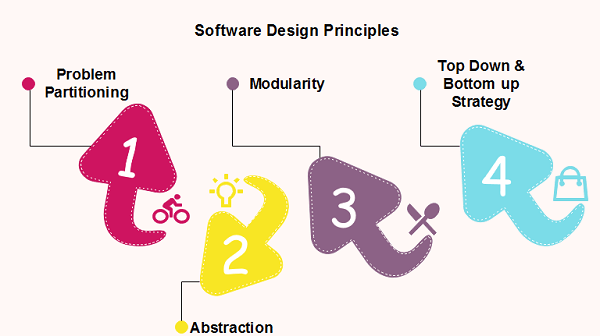
Requirement engineering is an important stage in any software development. It is also believed if the requirement engineering is done appropriately then we can expect that the final software developed doesn’t lag behind in terms of design or functionality leading to successful and profitable software.

1. The requirement engineering provides a vision of the final software i.e. what the software would do? This creates a sense of mutual understanding between the customer and the software developer.
2. Requirement engineering also helps in defining the scope of the software i.e. what will be the functionalities of the final software.
3. It also helps in perceiving the cost of the final software.
4. It also helps in perceiving the schedule up to which the software will be delivered to the customer.

**Software Design Principles:**

**Answers:**

Software design principles are concerned with providing means to handle the complexity of the design process effectively. Effectively managing the complexity will not only reduce the effort needed for design but can also reduce the scope of introducing errors during design.

****

**Explain the concept of modularity in software design. How does it improve maintainability and scalability of software systems?**

**Answer:**

Modularity in software design is a technique where complex software is divided into smaller, independent modules, such as functions, classes, or components. It facilitates easier management and understanding of complex systems by breaking them down into digestible parts.

Ease of Reading: Software becomes more readable as it's separated into specific functions handling distinct aspects of the overall functionality.

Simpler Testing: Testing is more straightforward and detailed when focusing on smaller, singular-function modules.

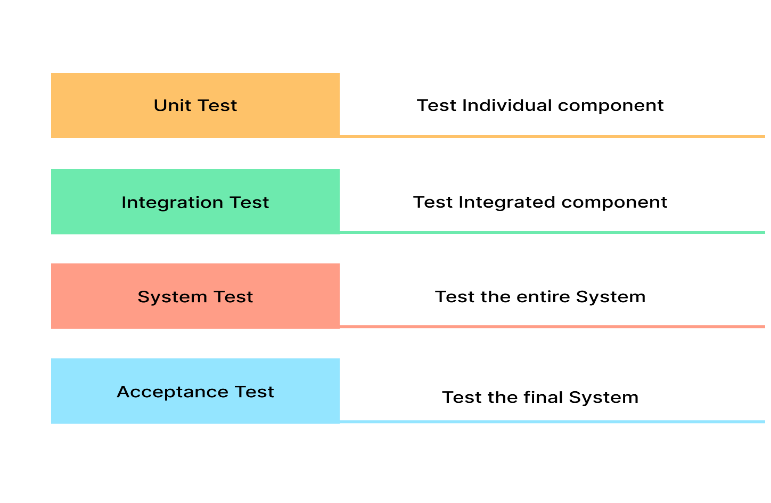
System Focus: Developers can concentrate on individual system parts without affecting the entire system, enhancing development efficiency.

Code Organization: Grouping similar functions into files and libraries makes finding specific code easier, streamlining development.

**Testing in Software Engineering**:

**Answer:**

Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases. The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability.



**Describe the different levels of software testing (unit testing, integration testing, system testing, acceptance testing). Why is testing crucial in software development?**

1. Unit/Component Testing:

Unit testing is done at the code level, where each component is tested individually to ensure their impartiality and analyze their functionality. Automating unit tests is possible and highly recommended in today’s fast-paced development environment. To make a unit test, you should outline what you expect the code to do and write the code, which will check if it is doing what you expect. You should then run the unit test to verify that everything works as expected. For example, let’s say you have a calculator program that adds two numbers together. You can create a unit test that verifies the numerical values that the calculator program returns are correct. You could also create tests that verify edge cases and errors are handled correctly.

In this simple example, you could use unit testing to verify that the calculator program adds two numbers correctly.

First, outline and document the expectations of the program, such as:

The program should accept two numerical values

It should return the sum of these two values

It should also handle negative numbers correctly

You can then write unit tests that feed these values into the program and verify the correct output. For example, you could have a test that checks the program returns 2 when 1 and 1 are entered. You could have another test that checks the program returns -3 when 1 and -4 are entered. Once all the tests have been written and executed, you can confidently say that the calculator program has been successfully tested and works as expected.

2. Integration Testing

Integration testing enables software testers to test group units integrated into a system or subsystems; it helps identify any bugs or issues arising from coding errors or integrations between modules. It is possible to automate integration testing.

3. System Testing

System testing is performed on an integrated environment comprising the whole application, where all components are assessed against specific business requirements. You can use automation tools for System Testing.

For example, Test sigma, a no-code test automation platform, can complete end-to-end flows for web, mobile, and desktop applications and APIs.

4. Acceptance Testing

Acceptance testing involves testing the system’s Functional and Non-functional aspects, such as performance, security, usability, accessibility, compatibility, and reliability. Depending on the system’s complexity, it can be done manually or through automation tools. In this example, we will demonstrate the process of using Test sigma to automate the acceptance testing of a login page. First, we must create a test scenario that simulates users entering their login credentials and logging in successfully. Testsigma will automatically detect any issues with the page and report them back to us. Using Testsigma for acceptance testing, we can ensure that our login page is working as expected and ready for deployment

**Version Control Systems:**

**What are version control systems, and why are they important in software development? Give examples of popular version control systems and their features.**

**Answer:**

Version control, also known as versioning or source control, is the practice of managing changes to source code. It’s about keeping a detailed account of every modification made to the code, ensuring that these changes are both trackable and reversible. A crucial tool in every developer’s kit, version control systems are the bedrock of this process.

**Why version control matters in software development**

Version control plays an integral role in both software development and project management. The tracking and versioning of source code changes in the repository form the heart of this process.

Let’s highlight three key reasons why version control is important to both software developers and product/project managers:

**Streamlined release management**

**Confliction prevention**

**Tracking changes to digital artifacts**

**Here’s a brief overview of some commonly used version control tools**

SVN — SVN, short for Subversion, is a centralized version control tool. While it’s more advanced than local version control, it carries a risk: if the central repository becomes corrupted, it’s difficult to retract the changes

TFS — Team Foundation Server (TFS) by Microsoft is also a centralized version control system, sharing the same limitations as SVN. When coupled with Azure DevOps, however, it can work with GIT to provide a distributed version control system

Mercurial — This open-source, distributed version control system is preferred over TFS and SVN because it mirrors the central repository and history in each local repository, preventing total loss of information and data corruption

Git — Git is a distributed repository version control system similar to Mercurial. It comes coupled with GitLab (a repository management software) and GitHub (a platform to upload copies of the repository)

**Software Project Management:**

**Discuss the role of a software project manager. What are some key responsibilities and challenges faced in managing software projects?**

**Answer:**

A project manager in software is responsible for leading a team of software developers and ensuring that software projects are completed on time, within budget, and to the satisfaction of the stakeholders. They are responsible for planning, executing, and closing projects. This involves defining project scope, creating schedules, allocating resources, managing risks, and monitoring progress.

The project manager is also responsible for communicating with stakeholders, including clients, management, and team members, to ensure everyone is on the same page. They must have strong leadership skills, be able to manage competing priorities, and have a deep understanding of software development methodologies and tools. Overall, the role of a project manager in software is critical to ensuring the success of software projects.

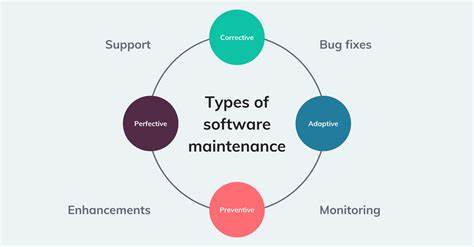


**Software Maintenance:**

**Define software maintenance and explain the different types of maintenance activities. Why is maintenance an essential part of the software lifecycle?**

**Answer:**

Software maintenance is the modification of a software product after delivery.



**Ethical Considerations in Software Engineering:**

**What are some ethical issues that software engineers might face? How can software engineers ensure they adhere to ethical standards in their work?**

**Answer:**

Ethics is a cornerstone in directing the revolution of technology, ensuring that technological progress aligns seamlessly with moral principles and societal values. In the realm of technology, numerous ethical considerations come to the fore, notably:

Safeguarding Human Values: Technology should uphold and amplify our core values like fairness, privacy, and accountability. Unforeseen consequences can lead to discrimination, manipulation, and power imbalances.

Building Trust and Transparency: Ethical considerations foster trust in technology, encouraging its responsible use. Transparency in algorithms, data handling, and decision-making processes is key.

Mitigating Risks and Harms: Unchecked technological advancements can threaten privacy, security, and the environment. Proactive ethical considerations help mitigate these harms and ensure responsible development.

**Ethical Dilemmas in Software Development**

Ethical dilemmas in software development arise when developers face situations where they must make decisions that may have conflicting ethical implications. These dilemmas can be challenging to navigate, as they often involve balancing competing values and considering the potential impact of decisions on individuals, society, and the environment.

**Real-Life Scenarios**

Ethics are essential in software development, guiding decisions that impact individuals, society, and the environment. Here are some real-life cases highlighting ethical considerations:

**Therac-25 Radiation Therapy Machine (1980s):** The Therac-25 incident from the ’80s, shed light on a crucial ethical concern. The software, created by developers, caused radiation overdoses. This highlighted a big ethical problem: developers must ensure their software is safe, especially in critical things like healthcare. It’s a reminder of the far-reaching consequences when ethical considerations aren’t prioritized in technology development.

**Equifax Data Breach (2017):** The Equifax data breach in 2017 revealed a significant ethical issue tied to developers. The breach happened because of security vulnerabilities in Equifax’s software. This case emphasizes the ethical duty of developers to prioritize strong security measures and safeguard sensitive user data. It’s a reminder of how crucial it is for developers to ensure systems are airtight, especially when handling people’s private information.

**Boeing 737 MAX Software Issues:** The Boeing 737 MAX crashes were tied to a flawed system, the Maneuvering Characteristics Augmentation System (MCAS). Ethical concerns arose about prioritizing profits over safety, insufficient training, and a lack of transparency in development. It’s a stark reminder of the importance of balancing safety and business interests in high

**REFERENCE**

<https://1.bp.blogspot.com/-jGx_g7sZOoQ/X4hBpV1JFLI/AAAAAAAACzo/v_CiPm6_L8sA8tbXJCVqInOk90nXi-kNwCLcBGAsYHQ/s16000/project%2Bmanager%2Bresponsibilities.png>

<https://www.javatpoint.com/software-engineering-software-design-principles>

<https://www.geeksforgeeks.org/version-control-systems/>

<https://en.wikipedia.org/wiki/Software_maintenance>