**Name: EKPO ENO-OBONG**

**SE\_Day1 QUESTIONS**

**August 2024**

Software Engineering Day1 Assignment

#Part 1: Introduction to Software Engineering

Explain what software engineering is and discuss its importance in the technology industry.

Identify and describe at least three key milestones in the evolution of software engineering.

List and briefly explain the phases of the Software Development Life Cycle.

Compare and contrast the Waterfall and Agile methodologies. Provide examples of scenarios where each would be appropriate.

Describe the roles and responsibilities of a Software Developer, a Quality Assurance Engineer, and a Project Manager in a software engineering team.

Discuss the importance of Integrated Development Environments (IDEs) and Version Control Systems (VCS) in the software development process. Give examples of each.

What are some common challenges faced by software engineers? Provide strategies to overcome these challenges.

Explain the different types of testing (unit, integration, system, and acceptance) and their importance in software quality assurance.

#Part 2: Introduction to AI and Prompt Engineering

Define prompt engineering and discuss its importance in interacting with AI models.

Provide an example of a vague prompt and then improve it by making it clear, specific, and concise. Explain why the improved prompt is more effective.

**ANSWERS**

Software engineering is the process of developing, testing and deploying computer applications to solve real-world problems by adhering to a set of engineering principles and best practices.

IMPORTANCE

Software engineering provides a way to manage the resources, schedule, and budget of the project, and to ensure that the team is working together effectively. Overall, software engineering is essential for creating high-quality software that meets the needs of the users and is easy to maintain.

**3 KEY MILESTONES IN SOFTWARE ENGINEERING**

1968: The first NATO Software Engineering Conference is held.

1970s: A number of new methodologies were developed, including structured programming and object-oriented programming.

1980s: The first computer-aided software engineering (CASE) tools were released

**SOFTWARE DEVELOPMENT PHASES**

The 7 Phases Of Software Development Life Cycle (SDLC)

Stage 1: Project Planning

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

Gathering Requirement and 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 analyze the requirements keeping the design and code of the software in mind. Further, investigating the validity and possibility of incorporating these requirements into the software system. The main goal of this stage is that everyone understands even the minute detail of the requirement. Hardware, operating systems, programming, and security are to name the few requirements.

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. Once the developer decides on the best design approach, he then selects the program languages like Oracle, Java, etc., that will suit the software.

Once the design specification is prepared, all the stakeholders will review this plan and provide their feedback and suggestions. It is absolutely mandatory to collect and incorporate stakeholder’s input in the document, as a small mistake can lead to cost overrun.

Stage 4: Coding or Implementation

Coding or Implementation: 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. This stage is considered to be one of the longest in SDLC. The developers need certain predefined coding guidelines, and programming tools like interpreters, compilers, debugger to implement the code.

The developers can show the work done to the business analysts in case if any modifications or enhancements required.

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.

After testing, the QA and testing team might find some bugs or defects and communicate the same with the developers. The development team then fixes the bugs and send it to QA for a re-test. This process goes on until the software is stable, bug-free and working according to the business requirements of that system.

Stage 6: Deployment

Once the testing is done, and the product is ready for deployment, 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. Again, a small round of testing is performed on production to ensure environmental issues or any impact of the new release.

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.

**Compare and contrast the Waterfall and Agile methodologies**.

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

**Describe the roles and responsibilities of a Software Developer, a Quality Assurance Engineer, and a Project Manager in a software engineering team**

Software Developer

The developer is responsible for writing the code and developing the entire software product. Aside from coding, a software developer is responsible for sending updates to the project manager and working closely with other team members, including the designer and QA engineer.

The kind of software developer you hire for your team depends on your desired software. If you want a mobile application, you need an iOS or an Android developer. However, if you want to develop a web application, you need a front-end, back-end or full-stack developer

Responsibilities

Developing applications, programs and systems using programming languages and frameworks.

Maintaining and updating software to keep it functional.

Collaborating with other team members to ensure best practices when developing software.

Report to the project manager about the progress of the software development.

QA Engineer

The QA or quality assurance engineer creates tests that identify issues with software before it is deployed.

QA engineers monitor every phase of the software development process, including development, testing, debugging and delivery. They ensure that quality is maintained at every stage of the development process and that the final product meets the requirements.

Responsibilities

Collaborate with stakeholders to understand and clarify software requirements.

Create development standards and procedures for the programmers to follow.

Confirm that the software meets the requirements before deployment.

Analyse the product to identify bugs and suggest changes to make them more efficient.

Develop and execute automation scripts using open-source tools

Project Manager

In software development, the roles of the project manager may include helping the team to design, execute, monitor and finish their work so that the clients will receive higher-quality software products.

Responsibilities

Discuss the project and its requirements with clients and software developers.

Assemble and lead the software development team.

Create the blueprint for the project, including the scope, allocating resources, setting deadlines and laying out communication strategies.

Supervising each stage of the software development project.

Set the budget and ensure the project adheres to it as closely as possible.

Tracking and communicating information regarding the project milestones, deliverables and change requests.

Deliver the completed software to the client and regularly check its performance.

**Discuss the importance of Integrated Development Environments (IDEs) and Version Control Systems (VCS) in the software development process. Give examples of each.**

\*\*An integrated development environment (IDE) is a software application that helps programmers develop software code efficiently. It increases developer productivity by combining capabilities such as software editing, building, testing, and packaging in an easy-to-use application IDE stands for Integrated Development Environment.

It is a lot of power tools bundled together in a way that they work together very productively.

An IDE will have an editor that not only highlights the syntax of your code, but also instantly identifies many types of errors and suggests fixes or what you might be intending to type next.

An IDE makes it easy to navigate through your code. The editor in the IDE should understand your code the same way the compiler understands the code.

An IDE has a way to keep your project organized. The IDE can build the entire project, ensuring everything is compiled and other steps taken to produce the final result.

An IDE will typically offer one or more source source code control systems so it is easy to synchronize and marge your code changes with the changes from other developers working on the same overall project.

An IDE will usually have a source level debugger that lets you set breakpoints, step through code, examine variables and data structures, making it easier to identify where is the problem (eg bug) you are trying to find

EXAMPLES OF IDE

NetBeans, Eclipse, IntelliJ, and Visual Studio etc

VERSION CONTROL SYSTEMS

Version control systems streamline development processes, enabling faster iteration and delivery of features. Efficient branching and merging capabilities allow developers to work concurrently on various aspects of a project without interference, significantly reducing the time from development to deployment

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.

IMPORTANCE

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. A separate branch is created for every contributor who made the changes and the changes aren’t merged into the original source code unless all are analyzed as soon as the changes are green signaled they merged to the main source code. It not only keeps source code organized but also improves productivity by making the development process smooth.

Basically Version control system keeps track on changes made on a particular software and take a snapshot of every modification. Let’s suppose if a team of developer add some new functionalities in an application and the updated version is not working properly so as the version control system keeps track of our work so with the help of version control system we can omit the new changes and continue with the previous version.

Benefits of the version control system:

Enhances the project development speed by providing efficient collaboration,

Leverages the productivity, expedites product delivery, and skills of the employees through better communication and assistance,

Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change,

Employees or contributors of the project can contribute from anywhere irrespective of the different geographical locations through this VCS,

For each different contributor to the project, a different working copy is maintained and not merged to the main file unless the working copy is validated.

Helps in recovery in case of any disaster or contingent situation,

Informs us about Who, What, When, Why changes have been made.

EXAMPLES OF VERSION CONTROL

Git , Subversion (SVN), Mercurial etc

**CHALLENGES OF SOFTWARE ENGINEERS**

ItThe rapid advancement of technology

For the IT sector, every technological innovation is a blessing. The rapid advancement of technology puts additional pressure on software development professionals to take advantage of these trends when creating new software products to stand out from the crowd and obtain a competitive advantage. It is one of the major software engineering problems.

Increasing customer demands in the development stage

The majority of software projects are conceptual in nature and are focused on creating software solutions that satisfy a range of consumer needs. Even the simplest application or product requires developers to fully grasp the underlying business concept and incorporate the necessary functionality to meet the ever-increasing client needs. It is among the software engineer coding challenges faced in software development.

Time limitation

The deadlines set for software engineers are incredibly short and are one of the major challenges of being a software engineer. After all, it is a game of time. When engineers collaborate with several clients across various time zones, the process becomes considerably more difficult. These time restraints frequently cause development teams to work less productively, resulting in subpar-quality products.

Limited infrastructure/ resources

The lack of resources or IT infrastructure to carry out projects successfully is another issue that most software development companies deal with and is among the major problems of software engineering. It could be caused by a lack of high-performance programming tools, strong computing platforms, ineffective data storage structures, or bad networks and connections. These obstacles lower software engineers' productivity and effectiveness, which affects the end result.

Understanding the large and complex system requirements is difficult

We are all aware that engineers do not communicate with clients directly because clients are contacted through the project manager or bidder procedure. As a result, engineers face challenges when dealing with customers who have personal ideas about the software they want to use because they rarely have direct contact with them. It is one of the key challenges in software engineering.

We all know that practically every development project requires pre-production and testing; therefore, the same problem arises when someone works on an unproven project. When working on a complicated project, managing your time and focusing on each component might be challenging.

Undefined system boundaries

There might not be a clear set of prerequisites for implementation. The customer might add additional unrelated and irrelevant functionalities on top of the crucial ones, which would result in a very high implementation cost that might go beyond the predetermined budget.

Customers are not clear about their needs

The lengthy list of features that clients frequently want in software may not always be clear to them. It may occur when they have a basic understanding of their requirements but haven't made many preparations for the execution phase.

Conflicting requirements

Two different project stakeholders can make expectations that are incompatible with one another's execution. A single customer may occasionally articulate two requirements that are incompatible.

Partitioning the system suitably to reduce complexity

Occasionally, the projects can be divided into smaller modules or functionalities, which are then handled by other teams. Larger, more complicated projects frequently call for additional segmentation, and the partitions must be kept separate from one another and without any overlap.

Validating and tracing requirements

Project requirements that are constantly changing make it harder for software engineers to work. Before beginning the implementation phase, it is crucial to double-check the list of requirements. Additionally, both forward and backward tracing should be possible.

Identifying critical requirements

It's crucial to identify the needs that must be fulfilled at all costs. Prioritizing the requirements will allow for the most urgent ones to be implemented first.

Resolving the "to be determined" requirements

Developers make sincere assumptions in the absence of specified specifications. They frequently avoid asking the product owners or customers their questions. Using what they know, they carry out the assignment. Later, they must recode the requirements using the information from the flaws.

Proper documentation, proper meetings time, and budget constraints

Confirm your grasp of the requirement by creating a clear requirement document. The aims, scope, limitations, and functional requirements of a product or software program are made clear to teams and individual stakeholders through documentation.

These are some of the major software engineer challenges, and approaches to deal with these have been discussed further. Check out the Best Full-Stack Developer Course online to gain knowledge in various aspects of full-stack development and be ready for a career in the field.

.Inadequate Testing and Debugging

Testing and debugging are important processes that must be performed on the developed software products to ensure that they meet the intended quality standards. But two of the main software engineer challenges here are that the practitioners often have time constraints such as shortage of time when developing the software and when testing it they are also likely to do it in an incomplete way due to shortage of time and resources.

Security and Privacy Concerns

As cybercrime and unauthorized access to personal data increase, the issues of security and privacy are pivotal to software engineering. Applications must now be fortified against such attacks, and designers need to include sophisticated access controls and data encryption into applications. These measures take time and are still susceptible to new age attacks, and thus require constant updates.

Scalability

Scalability is the measure of a software application’s capability to manage increased volumes of data as well as traffic. Scalability is a critical factor to consider for applications with lots of users and data volumes. But large scale applications that are tightly coupled with each and having dependencies are not easy to scale out in a short span of time.

High Availability

As for many core business applications, the level of availability must come very close to 100%. But, such reliable availability requires designing redundancy solutions, backups, and general failover capabilities to ensure that systems stay on in case of hardware melt downs or outage conditions.

It becomes crucial for engineers to pre-define the backup and failover mechanisms together with other disaster recovery procedures that run without any interruption and data loss to avoid any software engineer challenges. This is difficult to do for complex enterprise systems under budget constraints and time pressures.

Usability

Usability is a measure of how easy it can be for an end-user to navigate and interact with a software application. But, interfaces and workflows, which are easy to use, need a lot of effort while making them and multiple A/B tests are done before deployment.

Due to this, engineers and testers need to be asked to use the product thoroughly before deployment. However, this trial-and-error method takes time and also has the potential to leave out some bugs in the process.

Rapid Technological Change

Software engineers must continually update their skills to keep pace with the rapid evolution of technology. New programming languages, frameworks, and tools frequently come up, making it challenging to keep on upgrading. This constant learning curve can be quite stressful and also time-consuming, requiring continuous learning.

Cross-Functional Collaboration

Effective collaboration with other departments, such as marketing, sales, and customer support, is often challenging. Miscommunication and differing priorities can lead to delays and misunderstandings. Engineers must conquer these dynamics to ensure the end product aligns with business goals and user needs, requiring strong interpersonal and communication skills.

STRATEGIES TO OVERCOME THE CHALLENGES

* Maintaining accurate records
* Attempting to understand from the viewpoint of stakeholders
* Establishing proper communication with stakeholders
* Recognizing conflicting requirements from the stakeholder side
* Creating informative and well-structured conversations with end consumers
* Performing proper market research and competitors' analysis

**SOFTWARE TESTING & IMPORTANCE IN QUALITY ASSURANCE**

.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, Testsigma, 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 Testsigma 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

IMPORTANCE

Software Testing is a necessary process. It guarantees the software does what it should and meets all the requirements. Skipping testing can lead to severe issues that can mess up the software’s usefulness and how it works.

**PART 2: Define prompt engineering and discuss its importance in interacting with AI models**.

Prompt engineering is the process where you guide generative artificial intelligence (generative AI) solutions to generate desired outputs. Even though generative AI attempts to mimic humans, it requires detailed instructions to create high-quality and relevant output. In prompt engineering, you choose the most appropriate formats, phrases, words, and symbols that guide the AI to interact with your users more meaningfully. Prompt engineers use creativity plus trial and error to create a collection of input texts, so an application's generative AI works as expected.

A prompt is a natural language text that requests the generative AI to perform a specific task. Generative AI is an artificial intelligence solution that creates new content like stories, conversations, videos, images, and music. It's powered by very large machine learning (ML) models that use deep neural networks that have been pretrained on vast amounts of data.

The large language models (LLMs) are very flexible and can perform various tasks. For example, they can summarize documents, complete sentences, answer questions, and translate languages. For specific user input, the models work by predicting the best output that they determine from past training.

However, because they're so open-ended, your users can interact with generative AI solutions through countless input data combinations. The AI language models are very powerful and don't require much to start creating content. Even a single word is sufficient for the system to create a detailed response.

That being said, not every type of input generates helpful output. Generative AI systems require context and detailed information to produce accurate and relevant responses. When you systematically design prompts, you get more meaningful and usable creations. In prompt engineering, you continuously refine prompts until you get the desired outcomes from the AI system.

IMPORTANCE

* Optimizing Model Performance: Well-designed prompts can significantly enhance the performance of AI models by guiding them to generate more accurate and relevant responses.
* Improving User Experience: For applications leveraging AI, user satisfaction is often tied to how effectively the AI understands and responds to user inputs. Effective prompt engineering ensures a more intuitive and satisfactory user experience.
* Minimizing Errors and Bias: Thoughtfully crafted prompts can help in reducing errors and mitigating biases in AI outputs, leading to more reliable and fair results.
* Enhancing Creativity and Innovation: By mastering prompt engineering, one can unlock the creative potential of AI, enabling it to generate novel ideas, solutions, and content that might not be immediately apparent through traditional approaches

**\*\*Provide an example of a vague prompt and then improve it by making it clear, specific, and concise. Explain why the improved prompt is more effective.**

Vague prompt: “I need a book to read.”

Effective: “Can you suggest a science fiction novel with strong female protagonists?”

Explanation: The first prompt is open-ended and could result in any kind of book recommendation. The second prompt is clear about the genre and character preference, ensuring that the recommendation will be tailored to the user’s specific interests.