Software Engineering

The need for software engineering arises because of higher rate of change in user requirement and environment on which the software is working. Why the need for Software Engineering?

1. **Large software**: when the size of software become high, engineering has to step in to give it scientific process.
2. **Scalability**: if software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.
3. **Cost**: as hardware industry show it skill and manufacturing lower down the prices of computers and electronics devices. The cost of software remain high of proper process is not adapted.
4. **Dynamic Nature**: the always growing and adapting nature of software depends on the environment upon which the user works. If the nature of software is always changing, new enhancement has to be done in the existing one. This is where software engineering is needed.
5. **Quality Management**: better process of software development provides better and quality software product.

State the key differences and similarities between the following;

i. Classical waterfall model and Prototyping
ii. Spiral and Iterative
iii. Evolutionary and Prototyping model

Classical waterfall and Prototyping

Differences:

Classical waterfall has no feedback path and no mechanism for error correction. While **Prototyping** has a feedback path and provides mechanism for error correction at each phase.

Similarities:

Both **Classical waterfall** and **Prototyping model** has different stage or phase of development.

Spiral and Iterative

Differences:

Spiral model handles large, complex and dynamic software projects that needs constant change and adaption. While **Iterative model** handles small, simple, and stable software projects with fixed and clear requirements.

Similarities:

Both **spiral model** and **iterative model** has four phase each for software development process.

Evolutionary and Prototyping model

Differences:

Evolutionary model is used in object-oriented software development because the system can be easily be portioned into units in terms of object. While **Prototyping model** is used when technical issues are unclear and user requirements are not complete.

Similarities:

Both **Evolutionary model** and **Prototyping model** has a working model of the system as a guide.

A good software most have an SRS document which serve a good way to develop large software product. Briefly explain any six (6) Properties of a good SRS document and (4) Problems without using SRS document

Ans: Properties of a good SRS document

1. **Concise:** an SRS document should be concise and unambiguous, consistent, and complete. Verbose and irrelevant description reduce readability and increase error.
2. **Structured:** an SRS document should be well-structured. A well-structured SRS document is easy to understand and modify.
3. **Black-Box view:** it should only specify what the system should do and refrain from stating how to do those. Which means that SRS document should specify the external behavior of the and not to discuss the implementation issues.
4. **Conceptual Integrity:** it must show conceptual integrity for easy understanding when reading it.
5. **Response to Undesired Event:** it should characterize acceptable response for undesired events. (system response to external conditions).
6. **Verifiable:** all requirements in the SRS document should be verifiable.

Problems without using SRS document

1. Without SRS document, the system will not be implemented according to customer needs.
2. Software developers would not know whether what they are developing is exactly what the customer require
3. Without SRS document, it would be difficult for maintenance engineers to understand the functionality of the system.
4. Without an SRS document, it would be difficult for user document writers to write proper users' manuals.

Outline the stages in Software Development Lifecycle

Ans:

1. Communication
2. Requirement Gathering
3. Feasibility Study
4. System Analysis
5. Software Design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operational and Maintenance
11. Disposition

State and explain the criteria used to judge the quality of software product.

Ans:

The quality of software product is judge by what if offers and must satisfy the following;

1. **Operational**: how the software works in operation in terms of **budget, usability, efficiency, correctness, functionality, dependability, security and safety.**
2. **Translational**: this is where the software is move from one platform to another (i.e. from windows to macOS). It is based on **portability, interoperability, reusability, and adaptability.**
3. **Maintenance**: this aspect is important as software has the capability to maintain itself in ever changing environment. This also depend on **modularity, maintainability, flexibility, and scalability.**

Many software development lifecycles fall in to series of model being proposed, briefly state and explain any three (3) of such models.

1. ***Classica Waterfall Model***: The classical waterfall model is intuitively the most obvious way of to develop a software. However, it is not a practical model in the sense that it cannot be to develop the actual software. Thus, the model is considered as the theoretically way of developing software.
2. ***Iterative Waterfall Model***: This model provides a feedback path for error correction as and when detected later in the phase. Though errors are inevitable, but is important to detect them in the same phase in which they occur. As a result, it reduces the effort to correct bugs.
3. ***Evolutionary Model***: this model also refers to as *successive version or incremental model*. At first, a working model is built. Subsequently it undergoes functional improvement and keep adding new function until the desired system is built.
4. ***Prototyping Model***: A prototype is a toy implementation of the system. A prototype has limited functional capabilities, low reliability and inefficient performance as compared to the actual software. It is usually built using shortcut and therefore refers to as the crude version of the actual system.

A Rapid Application Development (RAD) model is based on prototyping and iterative model with no or less specific planning.

i. Explain prototyping and iteration

Prototyping: A prototype is a toy implementation of the system. This means a smaller version of the system with minimal functional capabilities.

Iteration: Is a model with working model of the system at a very early development where functional and design flaws are corrected.

ii. State (only) the phase in each model in (i).

Prototyping phases are;

- Planning
- Analysis
- Design
- System Prototype
- Implementation
- system

Iteration phases are;

- Requirement Specification
- System Design/Software Design
- Implementation and Unit Testing
- Integration and System Testing
- Operation and Maintenance

Software design is the first level to SDLC (Software Development Life Cycle) which moves the concentration from problem domain to solution domain. Briefly explain Software Design Level.

Ans:

1. **Architecture Design:** The architecture design is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. This level is where the designers get the ideas for the proposed solution domain.
2. **High-level Design:** High-level design focuses on how the system along with all of its component can be implemented in the form modules.
3. **Detailed Design:** this deal with the implementation part of what is seen as a system and its sub-system in the previous two designs. It defines logical structure of each module and their interfaces to communicate with each other.

Object oriented design strategy focuses on entities and its characteristic. The whole concept of software solution revolves around the engaged entities, state and explain the important concepts of Object-Oriented Design.

1. **Objects -** All entities involved in the solution design are known as objects. For example, person, banks, treated as objects.
2. **Classes -** A class is a generalized description of an object which is an instance of a class. Class defines all the attributes an object can have and methods defines the functionality of the object.
3. **Encapsulation -** In OOP, the attributes (data variables) and methods (operation on the data) are bundled together is called encapsulation. It restricts access of the data and methods from the outside world.
4. **Inheritance** - This makes it easier to define specific class and to create generalized classes from specific ones.
5. **Polymorphism -** OOP languages provide a mechanism where methods performing similar tasks but vary in arguments, can be assigned same name.

What are the steps involving in design process?

Ans:

- A solution design is created from requirement or previous used system.
- Objects are identified and grouped into classes on behalf of similarity in attribute characteristics.
- Class hierarchy and relation among them are defined.
- Application framework is defined.

Software design approaches has two generic approaches for software designing. State and explain the two approaches.

Ans:

1. **Top-down design-** This design takes the whole software system as one entity and then decomposes it to more sub-system based on some characteristics. Each subsystem is then treated as a system and decomposed further. This process keeps on running until the lowest level of system in the top-down hierarchy is achieved.
2. **Bottom-up Design -** The bottom-up design model starts with most specific and basic components. It proceeds with composing higher level of components by using basic or lower-level components. It keeps creating higher level components until the desired system is not evolved as one single component. With each higher level, the amount of abstraction is increased.

Explain the advantages of software reuse in software development and state any five (5) component that can be reuse.

Ans:

Advantages of software reuse

Software products are expensive. Software project managers are worried about the high cost of software development and are desperately look for ways to cut development cost. A possible way to reduce development cost is to reuse parts from previously developed software. In addition to reduced development cost and time, reuse also leads to higher quality of the developed products since the reusable components are ensured to have high quality.

Five (5) component that can be reuse are;

- Requirements specification
- Design
- Code
- Test cases
- Knowledge

Briefly explain the basic issues in any reuse program with reference to software development.

Ans:

Component creation- For component creation, the reusable components have to be first identified. Selection of the right kind of components having potential for reuse is important.

Component indexing and storing- Indexing requires classification of the reusable components so that they can be easily searched when looking for a component for reuse. The components need to be stored in a Relational Database Management System (RDBMS) or an Object-Oriented Database System (ODBMS) for easy access when the number of components becomes large.

Component searching- The programmers need to search for right components matching their requirements in a database of components. To be able to search

components efficiently, the programmers require a proper method to describe the components to be searched for.

Component understanding- The programmers need a precise and sufficiently complete understanding of what the component does to be able to decide whether they can reuse the component.

Component adaptation- Often, the components may need adaptation before they can be reused, since a selected component may not exactly fit the problem at hand.

Repository maintenance- A component repository once is created requires continuous maintenance. New components, as and when created have to be entered into the repository.

The faulty components have to be tracked.

The goal of system testing is to ensure that the develop system conforms to it requirement laid out the SRS document. Explain the three (3) kind of testing in software development process.

Ans:

- α – testing: It is the system testing performed by the development team.
- β –testing: It is the system testing performed by a friendly set of customers.
- Acceptance testing: It is the system testing performed by the customer himself after the product delivery to determine whether to accept or reject the delivered product.

Maintenance is key in any software developed project because no software is one hundred percent autonomous and correct. Kindly outline any three (3) activities of software maintenance with regard software.

Ans:

- Correcting errors that were not discovered during the product development phase. This is called **corrective maintenance.**
- Improving the implementation of the system, and enhancing the functionalities of the system according to the customer's requirements. This is called **perfective maintenance.**
- Porting the software to work in a new environment (i.e. working on a new computer platform or operating system. This is called **adaptive maintenance.**