Software Enginnering Assignment

Abhishek Tyagi
EN18CS301009
CS-A

Q.1) Explain in brief the impact of software engineering

Ans.1).

1. **Reduces complexity:** Software engineering divides big problems into various small issues. And then start solving each small issue one by one. All these small problems are solved independently to each other.
2. **To minimize software cost**: A lot of manpower is required to develop software with a large number of codes. But in software engineering, programmers project everything and decrease all those things that are not needed. In turn, the cost for software productions becomes less as compared to any software that does not use software engineering method.
3. **To decrease time:** Anything that is not made according to the project always wastes time. And if you are making great software, then you may need to run many codes to get the definitive running code. So if you are making your software according to the software engineering method, then it will decrease a lot of time.
4. **Handling big projects**: Since problems are broken into chunks for easy execution .So to handle a big project without any problem, the company has to go for a software engineering method.
5. **Effectiveness:** Effectiveness comes if anything has made according to the standards. Since it is automated process.

Q.2) Differentiate between functional and non-functional requirements.

Ans.2)

FUNCTIONAL REQUIREMENTS	NON FUNCTIONAL REQUIREMENTS
A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
It specifies “What should the software system do?”	It places constraints on “How should the software system fulfill the functional requirements?”
Functional requirement is specified by User.	Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers.
It is mandatory.	It is not mandatory.
It is captured in use case.	It is captured as a quality attribute.
Defined at a component level.	Applied to a system as a whole.
Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Functional Testing like System, Integration, End to End, API testing, etc are done.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done.
Usually easy to define.	Usually more difficult to define.

Q.3) What are the advantages and disadvantages of Waterfall model?

Ans.)

Waterfall Model - **Advantages**

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting,and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major Advantages of the Waterfall Model are as follows –

• Simple and easy to understand and use

• Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

• Phases are processed and completed one at a time.

• Works well for smaller projects where requirements are very well understood.

• Clearly defined stages.

• Well understood milestones.

• Easy to arrange tasks.

• Process and results are well documented.

Waterfall Model - **Disadvantages**

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows –

• No working software is produced until late during the life cycle.

• High amounts of risk and uncertainty.

• Not a good model for complex and object-oriented projects.

• Poor model for long and ongoing projects.

• Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.

• It is difficult to measure progress within stages.

• Cannot accommodate changing requirements.

• Adjusting scope during the life cycle can end a project.

• Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

Q.4) What is reusability in software engineering?

Ans.4) In computer science and software engineering, reusability is the use of existing assets in some form within the software product development process these assets are products and by-products of the software development life cycle and include code, software components, test suites, designs and documentation. The opposite concept of reusability is leverage, which modifies existing assets as needed to meet specific system requirements. Because reuse implies the creation of a separately maintained version of the assets, it is preferred over leverage.

Subroutines or functions are the simplest form of reuse. A chunk of code is regularly organized using modules or namespaces into layers.

The ability to reuse relies in an essential way on the ability to build larger things from smaller parts, and being able to identify commonalities among those parts. Reusability is often a required characteristic of platform software. Reusability brings several aspects to software development that do not need to be considered when reusability is not required.
Resusablitly increases efficiency.

Q.5) Explain the software requirement analysis and specification. Discuss various methods for requirement gathering.

Ans.5) Requirement analysis is significant and essential activity after elicitation. We analyze, refine, and scrutinize the gathered requirements to make consistent and unambiguous requirements.

This activity reviews all requirements and may provide a graphical view of the entire system.

i) **Draw the context diagram**: The context diagram is a simple model that defines the boundaries and interfaces of the proposed systems with the external world. It identifies the entities outside the proposed system that interact with the system.

(ii) **Development of a Prototype** (optional): One effective way to find out what the customer wants is to construct a prototype, something that looks and preferably acts as part of the system they say they want.

We can use their feedback to modify the prototype until the customer is satisfied continuously. Hence, the prototype helps the client to visualize the proposed system and increase the understanding of the requirements. When developers and users are not sure about some of the elements, a prototype may help both the parties to take a final decision.

The prototype should be built quickly and at a relatively low cost. Hence it will always have limitations and would not be acceptable in the final system. This is an optional activity.

(iii) **Model the requirements**: This process usually consists of various graphical representations of the functions, data entities, external entities, and the relationships between them. The graphical view may help to find incorrect, inconsistent, missing, and superfluous requirements.

Such models include the Data Flow diagram, Entity-Relationship diagram, Data Dictionaries, State-transition diagrams, etc.

(iv) **Finalise the requirements**: After modeling the requirements, we will have a better understanding of the system behavior. The inconsistencies and ambiguities have been identified and corrected. The flow of data amongst various modules has been analyzed. Elicitation and analyze activities have provided better insight into the system. Now we finalize the analyzed requirements, and the next step is to document these requirements in a prescribed format.

Software Requirement Specifications

The production of the requirements stage of the software development process is Software Requirements Specifications (SRS) (also called a requirements document). This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed.

First, the SRS could be written by the client of a system. Second, the SRS could be written by a developer of the system. The first case, SRS, is used to define the needs and expectation of the users. The second case, SRS, is written for various purposes and serves as a contract document between customer and developer.

Methods describe how tasks are performed under specific circumstances. A method may have none or one or more related techniques. A method should be related to at least one task.
The following are some of the well-known **requirements gathering methods** –

Brainstorming

Brainstorming is used in requirement gathering to get as many ideas as possible from group of people. Generally used to identify possible solutions to problems, and clarify details of opportunities.

Document Analysis

Reviewing the documentation of an existing system can help when creating AS–IS process document, as well as driving gap analysis for scoping of migration projects. I Nuggets of information are often buried in existing documents that help us ask questions as part of validating requirement completeness.

Focus Group

A focus group is a gathering of people who are representative of the users or customers of a product to get feedback. The feedback can be gathered about needs/opportunities/ problems to identify requirements, or can be gathered to validate and refine already elicited requirements.

Interface analysis

. Interface analysis – reviewing the touch points with other
external systems is important to make sure we don't overlook requirements that aren't immediately visible to users.

Interview

Interviews of stakeholders and users are critical to creating the great software. Without understanding the goals and expectations of the users and stakeholders, we are very unlikely to satisfy them.

Observation

By observing users, an analyst can identify a process flow, steps, pain points and opportunities for improvement. Observations can be passive or active (asking questions while observing).

Prototyping

In this approach, you gather preliminary requirements that you use to build an initial version of the solution – a prototype. You show this to the client, who then gives you additional requirements.

Requirement Workshops

More structured than a brainstorming session, involved parties collaborate to document requirements. A workshop will be more effective with two analysts than with one.

Reverse Engineering

When a migration project does not have access to sufficient documentation of the existing system, reverse engineering will identify what the system does. It will not identify what the system should do, and will not identify when the system does the wrong thing.

Survey/Questionnaire

When collecting information from many people – too many to interview with budget and time constraints – a survey or questionnaire can be used. The survey can force users to select from choices, rate something ("Agree Strongly, agree..."), or have open ended questions allowing free-form responses. Survey design is hard – questions can bias the respondents.