**Name :-MD.Tanviralam**

**ID:- 2119170081**

**Course Title:- Software Engineering**

**Course Code:- CSI 321**

**Department :- BSc in CSE**

**Semester:- Summer-2022**

**Ans to the Q no :- 01(a)**

Software engineering is the application of principles used in the field of engineering, which usually deals with physical systems, to the design, development, testing, deployment and management of software systems.

**Ans to the Q no :- 01(b)**

**Definition:** The waterfall model is a classical model used in system development life cycle to create a system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion. This model is divided into different phases and the output of one phase is used as the input of the next phase. Every phase has to be completed before the next phase starts and there is no overlapping of the phases.

**Description:** The sequential phases described in the Waterfall model are:

1. Requirement Gathering- All possible requirements are captured in product requirement documents.

2. Analysis Read - the requirement and based on analysis define the schemas, models and business rules.

3. System Design -- Based on analysis design the software architecture.

4. Implementation Development of the software in the small units with functional testing.

5. Integration and Testing Integrating of each unit developed in previous phase and post integration test the entire system for any faults.

6. Deployment of system - Make the product live on production environment after all functional and nonfunctional testing completed.

7. Maintenance Fixing issues and release new version with the issue patches as required.

**Advantages:** 1. Easy to use, simple and understandable,

2. Easy to manage as each phase has specific outputs and review process,

3. Clearly-defined stages,

4. Works well for smaller projects where requirements are very clear,

5. Process and output of each phase are clearly mentioned in the document.

**Disadvantages:**

1. It doesn’t allow much reflection or revision. When the product is in testing phase, it is very difficult to go back and change something which is left during the requirement analysis phase.

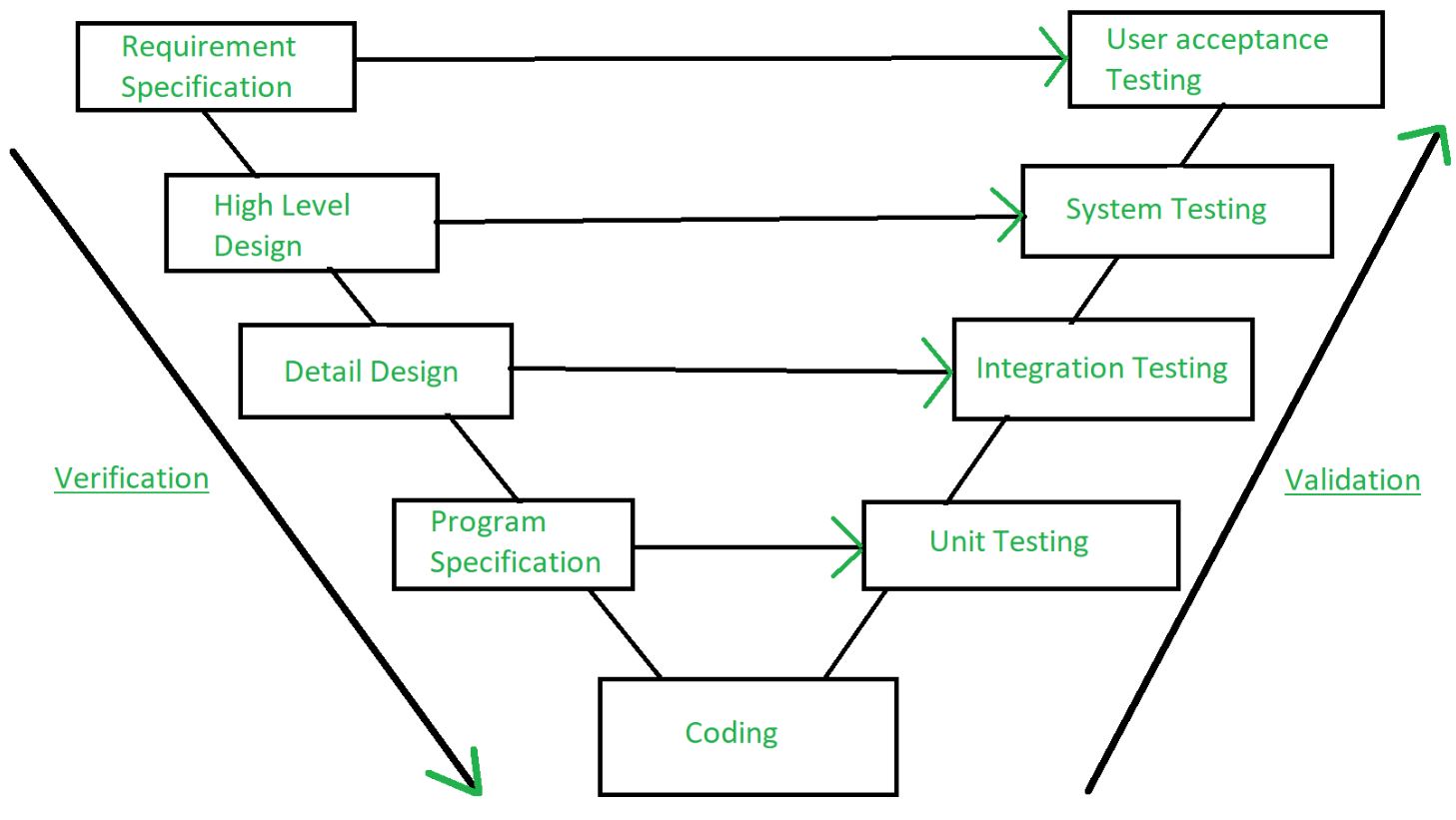
2. Risk and uncertainty are high.

3. Not advisable for complex and object-oriented projects.

4. Changing requirements can’t be accommodated in any phase.

5. As testing is done at a later phase. So, there is a chance that challenges and risks at earlier phases are not identified.

**Validation process**:-Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product.



**Ans to the Q no :- 01(c)**

Software requirement specification (SRS) is a document that completely describes what the proposed software should do without describing how software will do it. The basic goal of the requirement phase is to produce the SRS, Which describes the complete behavior of the proposed software. SRS is also helping the clients to understand their own needs.

**Characteristics of an SRS**

• Correct

• Complete

• Unambiguous

• Verifiable

• Consistent

• Ranked for importance and/or stability

• Modifiable

• Traceable

An SRS is correct if every requirement included in the SRS represents something required in the final system. An SRS is complete, if everything the software is supposed to do and the responses of the software to all classes of input data are specified in the SRS. Correctness ensures that what is specified is done correctly, completeness ensures that everything is indeed specified. An SRS is unambiguous if and only if every requirement stated has one and only one interpretation. Requirements are often written in natural language, which are inherently ambiguous.

An SRS is verifiable if and only if every stated requirement is verifiable. A requirement is verifiable if there exists some cost-effective process that can check whether the final software meets that requirement. An SRS is consistent if there is no requirement that conflicts with another.

Terminology can cause inconsistencies; for example, different requirements may use different terms to refer to the same object. All the requirements for software are not of equal importance. Some are critical, others are important but not critical, and there are some, which are desirable, but not very important. An SRS is ranked for importance and the stability of the requirement are indicated. Stability of requirement reflects the chances of it changing in future. An SRS is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development. Forward traceability means that each requirement should be traceable to some design and code elements. Backward traceability requires that it be possible to trace design and code elements to the requirements they support. Traceability aids verification and validation.

**Ans to the Q no :- 02(a)**

Categories of Software Maintenance:-

The four different types of software maintenance are each performed for different reasons and purposes. A given piece of software may have to undergo one, two, or all types of maintenance throughout its lifespan.

**The four types are:**

1.Corrective Software Maintenance

2.Preventative Software Maintenance

3.Perfective Software Maintenance

4.Adaptive Software Maintenance

**1.Corrective Software Maintenance**

Corrective software maintenance is the typical, classic form of maintenance (for software and anything else for that matter). Corrective software maintenance is necessary when something goes wrong in a piece of software including faults and errors. These can have a widespread impact on the functionality of the software in general and therefore must be addressed as quickly as possible.

Many times, software vendors can address issues that require corrective maintenance due to bug reports that users send in. If a company can recognize and take care of faults before users discover them, this is an added advantage that will make your company seem more reputable and reliable (no one likes an error message after all).

**2.Preventative Software Maintenance**

Preventative software maintenance is looking into the future so that your software can keep working as desired for as long as possible.

This includes making necessary changes, upgrades, adaptations and more. Preventative software maintenance may address small issues which at the given time may lack significance but may turn into larger problems in the future. These are called latent faults which need to be detected and corrected to make sure that they won’t turn into effective faults.

**3.Perfective Software Maintenance**

As with any product on the market, once the software is released to the public, new issues and ideas come to the surface. Users may see the need for new features or requirements that they would like to see in the software to make it the best tool available for their needs. This is when perfective software maintenance comes into play.

4**. Adaptive Software Maintenance**

Adaptive software maintenance has to do with the changing technologies as well as policies and rules regarding your software. These include operating system changes, cloud storage, hardware, etc. When these changes are performed, your software must adapt in order to properly meet new requirements and continue to run well.

**Ans to the Q no :- 02(b)**

Difference between Black box testing and white box testing:-

|  |  |
| --- | --- |
| **Black Box Testing** | **White Box Testing** |
| 1. It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it. | 1. It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software. |
| 2. Implementation of code is not needed for black box testing. | 2. Code implementation is necessary for white box testing. |
| 3. It is mostly done by software testers. | 3. It is mostly done by software developers. |
| 4. No knowledge of implementation is needed. | 4. Knowledge of implementation is required. |
| 5. It can be referred to as outer or external software testing. | 5. It is the inner or the internal software testing. |

**Ans to the Q no :- 02(C)**

The plan identifies the SQA responsibilities of a team, and lists the areas that need to be reviewed and audited. It also identifies the SQA work products.

The SQA plan document consists of the below sections:

1.Purpose section

2.Reference section

3.Software configuration management section

4.Problem reporting and corrective action section

5.Tools, technologies, and methodologies section

Code control section

6.Records: Collection, maintenance, and retention section

7.Testing methodology

8.SQA Activities

**Given below is the list of SQA activities:**

**1) Creating an SQA Management Plan:**

The foremost activity includes laying down a proper plan regarding how the SQA will be carried out in your project.

Along with what SQA approach you are going to follow, what engineering activities will be carried out, and it also includes ensuring that you have the right talent mix in your team.

**2) Setting the Checkpoints:**

The SQA team sets up different checkpoints according to which it evaluates the quality of the project activities at each checkpoint/project stage. This ensures regular quality inspection and working as per the schedule.

**3) Apply software Engineering Techniques:**

Applying some software engineering techniques aids a software designer in achieving high-quality specifications. For gathering information, a designer may use techniques such as interviews and FAST (Functional Analysis System Technique).

Later, based on the information gathered, the software designer can prepare the project estimation using techniques like WBS (work breakdown structure), SLOC (source line of codes), and FP(functional point) estimation.

**4) Executing Formal Technical Reviews:**

An FTR is done to evaluate the quality and design of the prototype.

In this process, a meeting is conducted with the technical staff to discuss the actual quality requirements of the software and the design quality of the prototype. This activity helps in detecting errors in the early phase of SDLC and reduces rework effort in the later phases.

**5) Having a Multi-Testing Strategy:**

By multi-testing strategy, we mean that one should not rely on any single testing approach, instead, multiple types of testing should be performed so that the software product can be tested well from all angles to ensure better quality.

**6) Enforcing Process Adherence:**

This activity insists on the need for process adherence during the software development process. The development process should also stick to the defined procedures.

**Ans to the Q no :- 03(A)**

Software Project Management (SPM) is a proper way of planning and leading software projects. It is a part of project management in which software projects are planned, implemented, monitored, and controlled. Need for Software Project Management: Software is a non-physical product.

**Ans to the Q no :- 03(B)**

**Project managers role**:- Project managers play the lead role in planning, executing, monitoring, controlling, and closing out projects. They are accountable for the entire project scope, the project team and resources, the project budget, and the success or failure of the project.

**Project manager responsibilities:-**

A project manager, with the help of their team, is charged with multiple responsibilities that span the five project phases of a project life cycle (initiating, planning, executing, monitoring, and closing) below.

The project management phases intersect with 10 knowledge areas, which include integration, scope, time, cost, quality, human resources, communication, risk procurement, and stakeholder management.

**Initiating phase**

1.Integration management: Developing a project charter

2.Stakeholder management: Identifying stakeholders

**Planning phase**

1.Integration management: Developing a project management plan

2.Scope management: Defining and managing scope, creating a work breakdown structure (WBS), and requirements gathering

3.Time management: Planning, defining, and developing schedules, activities, estimating resources and activity durations

4.Costs management: Planning and estimating costs, and determining budgets

5.Quality management: Planning and identifying quality requirements

6.Human Resource management: Planning and identifying human resource needs

7.Communications management: Planning communications

8.Risk management: Planning for and identifying potential risks, performing qualitative and quantitative risk analysis, and planning risk mitigation strategies

9.Procurement management: Planning for and identifying required procurements

10.Stakeholder management: Planning for stakeholder expectations

**Executing**

1.Integration management: Directing and managing all work for the project

2.Quality management: Performing all aspects of managing quality

3.Human resource management: Selecting, developing, and managing the project team

4.Communications management: Managing all aspects of communications

5.Procurement management: Take action on securing necessary procurements

6.Stakeholder management: Managing all stakeholder expectations

**Monitoring and controlling**

1.Integration management: Monitoring and controlling the project work and managing any necessary changes

2.Scope management: Validating and controlling the scope of the project

3.Time management: Controlling the scope of the project

4.Costs management: Controlling project costs

5.Quality management: Controlling the quality of deliverables

**Closing**

1.Integration management: Closing all phases of the project

2.Procurement management: Closing all project procurements

**Ans to the Q no :- 03(C)**

Iterative Model advantage and disadvantage:-

**Advantages of Iterative model:**

1.In iterative model we can only create a high-level design of the application before we actually begin to build the product and define the design solution for the entire product. Later on we can design and built a skeleton version of that, and then evolved the design based on what had been built.

2.In iterative model we are building and improving the product step by step. Hence we can track the defects at early stages. This avoids the downward flow of the defects.

3.In iterative model we can get the reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.

4.In iterative model less time is spent on documenting and more time is given for designing.

**Disadvantages of Iterative model:**

1. Each phase of an iteration is rigid with no overlaps

2.Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle.

**Ans to the Q no :- 04(A)**

**Quality**:-In the context of software engineering, software quality measures how well the software is designed (quality of design), and how well the software conforms to that design (quality of conformance). It is often described as the 'fitness for purpose' of a piece of software.”

**Importance of Quality**:-Quality assurance is the process of managing the quality of the software throughout the development process. It is undeniably the most important step within an organization’s development process. Not only does it help to prevent mistakes and defects to the software being developed, it also ensures that customers receive the highest quality products, services and solutions. In short, quality assurance ensures accuracy and is the deciding factor for your product launch.

Although quality assurance, quality control and software testing services are sometimes used interchangeably, they are different. Quality assurance focuses on preventative activities, while quality assurance focuses on corrective actions. Quality control is a subset of quality assurance, while software testing is a subset of quality control.

**Quality assurance** is the procedure of examining the software development process to prevent possible defects and ensure the quality of products, services and solutions. Quality assurance is process oriented, meaning that it focuses on processes and procedures rather than conducting actual testing on the system. Process definition, process implementation, and audits are examples of quality assurance.

**Quality control** is the procedure of examining products, services or solutions to find defects and ensure quality of the end product. Other than quality assurance, it does not deal with the processes used to create a product, service or solution. It focuses on actual testing by executing the software with the aim to identify defects. Quality control is product oriented. Software testing, technical reviews, and code inspections are three examples of quality control. Quality control is a subset of quality assurance.

**Software testing** is the procedure of testing products, services or solutions before the software will be released. Other than quality control, software testing is a preventive process rather than a corrective process. Performance testing, security testing, usability testing are some of the examples of software testing. Software testing is a subset of quality control.

**Ans to the Q no :- 04(b)**

Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.

Software Quality Assurance is a process which works parallel to development of software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

**Software Quality Assurance has:**

1.A quality management approach

2.Formal technical reviews

3.Multi testing strategy

4.Effective software engineering technology

5.Measurement and reporting mechanism

Major Software Quality Assurance Activities:

SQA Management Plan:

Make a plan for how you will carry out the sqa through out the project. Think about which set of software engineering activities are the best for project. check level of sqa team skills.

Set The Check Points:

SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.

Multi testing Strategy:

Do not depend on a single testing approach. When you have a lot of testing approaches available use them.

Measure Change Impact:

The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to change check the compatibility of this fix with whole project.

Manage Good Relations:

In the working environment managing good relations with other teams involved in the project development is mandatory. Bad relation of sqa team with programmers team will impact directly and badly on project. Don’t play politics.