SOLUTION-2019

Q.1 Answer the following

a) Distinguish between a program and a software.
b) Without preparing the SRS document, an organization might face several problems. (True/False) Justify.
c) List the advantage and disadvantage of prototype life cycle model.
d) What do you mean by coftrol flow based design? Discuss with an exaniple.
e) What is debugging?

(SAGNIK MISRA 1729152)

A) Programs are developed by individuals for their personal use. They are generally, small in size and have limited functionality. The author of a program himself uses and maintains his program, these usually do not have a good user interface and lack of proper documentation.
Whereas Software products have multiple users and therefore should have a good user interface, proper operating procedures, and good documentation support.Since a Software product has a large no of users, it must be properly designed, carefully implemented and properly tested.

B) True.This document lays a foundation for software engineering activities and is created when entire requirements are elicited and analyzed. SRS is a formal document, which acts as a representation of software that enables the users to review whether it (SRS) is according to their requirements. In addition, it includes user requirements for a system as well as detailed specifications of the system requirements.

C) Advantages:
1. The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
2. New requirements can be easily accommodated as there is scope for refinement.

Disadvantages:
1. There may be too much variation in requirements each time the prototype is evaluated by the customer.
2. Poor Documentation due to continuously changing customer requirements.

D) A control flow diagram helps us understand the detail of a process. It shows us where control starts and ends and where it may branch off in another direction, given certain situations.
Example- Let's say you are working on software to start a machine. What happens if the engine is flooded, or a spark plug is broken. Control then changes the flow to other parts of the software. We can represent these branches with a diagram. The flow diagram is helpful because it can be understood by both stakeholders and systems professionals. Although some of the symbols might not be fully understood by the layperson, they can still grasp the general concept.

E) Debugging is the process of detecting and removing of existing and potential errors in a software code that can cause it to behave unexpectedly or crash. To prevent incorrect operation of a software or system, debugging is used to find and resolve bugs or defects. When various subsystems or modules are tightly coupled, debugging becomes harder as any change in one module may cause more bugs to appear in another. Sometimes it takes more time to debug a program than to code it.

Q2 (a)List the major responsibilities of Software Project Manager.
(SAGNIK SARBADHIKARI 1729151)

The major responsibilities of Software Project Manager are:

A) Activity and resource planning
B) Organizing and motivating a project team
C) Controlling time management
D) Cost estimating and developing the budget
E) Ensuring customer satisfaction
F) Analyzing and managing project risk
G) Monitoring progress
H) Managing reports and necessary documentation

(b)What do you mean by agile approach? Explain how it is feasible to software life cycle. Explain with each component.

The Agile Method ensures that value is optimized throughout the development process. The use of iterative planning and feedback results in teams that can continuously align a delivered product that reflects the desired needs of a client.

Agile approach is feasible to software life cycle as:

1. Scope out and prioritize projects

During the first step of the agile software development life cycle, the team scopes out and prioritizes projects. Some teams may work on more than one project at the same time depending on the department's organization.

2. Diagram requirements for the initial sprint

Once you have identified the project, work with stakeholders to determine requirements. You might want to use user flow diagrams or high-level UML diagrams to demonstrate how the new feature should function and how it will fit into your existing system.

3. Construction/iteration

Once a team has defined requirements for the initial sprint based on stakeholder feedback and requirements, the work begins. UX designers and developers begin work on their first iteration of the project, with the goal of having a working product to launch at the end of the sprint.

4. Release the iteration into production

You're nearly ready to release your product into the world. Finish up this software iteration with the following steps:

- Test the system. Your quality assurance (QA) team should test functionality, detect bugs, and record wins and losses.
- Address any defects.
- Finalize system and user documentation. Lucidchart can help you visualize your code through UML diagrams or demonstrate user flows so everyone understands how the system functions and how they can build upon it further.
- Release the iteration into production.

5. Production and ongoing support for the software release

This phase involves ongoing support for the software release. In other words, your team should keep the system running smoothly and show users how to use it. The production phase ends when support has ended or when the release is planned for retirement.

6. Retirement

During the retirement phase, you remove the system release from production, typically when you want to replace a system with a new release or when the system becomes redundant, obsolete, or contrary to your business model.

Q.3 a) List five desirable characteristics of a good Software Requirements Specification (SRS) document.

(TUNEER BHATTACHARYA 1729091)

Following are the characteristics of a good SRS document:

1. **Correctness:**
   User review is used to ensure the correctness of requirements stated in the SRS. SRS is said to be correct if it covers all the requirements that are actually expected from the system.
2. **Completeness:**
   Completeness of SRS indicates every sense of completion including the numbering of all the pages, resolving the to be determined parts to as much extent as possible as well as covering all the functional and non-functional requirements properly.
3. **Consistency:**
   Requirements in SRS are said to be consistent if there are no conflicts between any set of requirements. Examples of conflict include differences in terminologies used at separate places, logical conflicts like time period of report generation, etc.
4. **Unambiguousness:**
   An SRS is said to be unambiguous if all the requirements stated have only 1 interpretation. Some of the ways to prevent unambiguousness include the use of modelling techniques like ER diagrams, proper reviews and buddy checks, etc.

5. **Ranking for importance and stability:**
   There should a criterion to classify the requirements as less or more important or more specifically as desirable or essential. An identifier mark can be used with every requirement to indicate its rank or stability.

b)When does the project planning activities start and end in software life cycle? List the important activities software project managers perform during project planning.

The project manager and project team have one shared goal: to carry out the work of the project for the purpose of meeting the project's objectives. Every project has a beginning, a middle period during which activities move the project toward completion, and an ending (either successful or unsuccessful). A standard project typically has the following four major phases (each with its own agenda of tasks and issues): initiation, planning, implementation, and closure. Taken together, these phases represent the path a project takes from the beginning to its end and are generally referred to as the project "life cycle.In the initiation phase of the project, you identify a business need, problem, or opportunity and brainstorm ways that your team can meet this need, solve this problem, or seize this opportunity. During this step, you figure out an objective for your project, determine whether the project is feasible, and identify the major deliverables for the project.Once your team has completed work on a project, you enter the closure phase. In the closure phase, you provide final deliverables, release project resources, and determine the success of the project. Just because the major project work is over, that doesn't mean the project manager's job is done—there are still important things to do, including evaluating what did and did not work with the project.

Q.4) a)Explain complete COCOMO model with advantages and disadvantages.

(TANAYA JAISWAL 1729088)

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry Boehm. The model uses a basic regression formula, with parameters that are derived from historical project data and current project characteristics.COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. The first level, Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors to account for difference in project attributes (Cost Drivers). Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases.

Advantages

1. COCOMO is factual and easy to interpret. One can clearly understand how it works.
2. Accounts for various factors that affect cost of the project.
3.Works on historical data and hence is more predictable and accurate.
4.The drivers are very helpful to understand the impact on the different factors that affect the project costs.

Disadvantages

1. COCOMO model ignores requirements and all documentation.
2. It ignores customer skills, cooperation, knowledge and other parameters.
3. It oversimplifies the impact of safety/security aspects.
4. It ignores hardware issues
5. It ignores personnel turnover levels
6. It is dependent on the amount of time spent in each phase.

b) What are the other software metric systems? Explain each of them and give their relevance with respect to COCOMO model.

Intermediate COCOMO computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes. This extension considers a set of four "cost drivers", each with a number of subsidiary attributes:-

- Product attributes
  - Required software reliability extent
  - Size of application database
  - Complexity of the product
- Hardware attributes
  - Run-time performance constraints
  - Memory constraints
  - Volatility of the virtual machine environment
  - Required turnabout time
- Personnel attributes
  - Analyst capability
  - Software engineering capability
  - Applications experience
  - Virtual machine experience
  - Programming language experience
- Project attributes
  - Use of software tools
  - Application of software engineering methods
  - Required development schedule

- The Intermediate COCOMO formula now takes the form:
- **$E=a_i(KLOC)^{(b_i)}(EAF)$**
- where E is the effort applied in person-months, **KLOC** is the estimated number of thousands of delivered lines of code for the project, and **EAF** is the factor calculated above. The coefficient $\mathbf{a}_i$ and the exponent $\mathbf{b}_i$ are given in the next table.

Q5 a) Write Short notes on one any two of the following

a) **Non-Functional Requirements**
b) **Waterfall model**
c) **Unit testing**

(SAYAHNA SENGUPTA 1729158)

A) NON FUNCTIONAL REQUIREMENTS

Basically, non-functional requirements describe how the system works.
The four examples of Non-Functional requirement groups:usability, reliability, performance, and supportability.

Usability:

Prioritize the important functions of the system based on usage patterns. Frequently used functions should be tested for usability, as should complex and critical functions. Be sure to create a requirement for this.

Reliability:

Users have to trust the system, even after using it for a long time.
Your goal should be a long MTBF (mean time between failures).
Create a requirement that data created in the system will be retained for a number of years without the data being changed by the system.
It's a good idea to also include requirements that make it easier to monitor system performance.

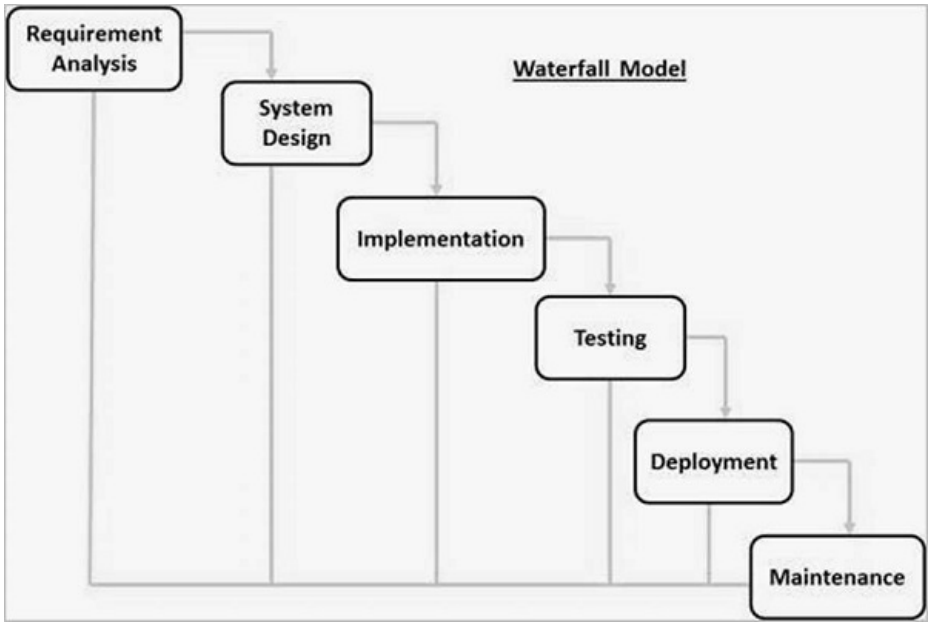
Performance: What should system response times be, as measured from any point, under what circumstances?Are there specific peak times when the load on the system will be unusually high?Think of stress periods, for example, at the end of the month or in conjunction with payroll disbursement.

Supportability:

The system needs to be cost-effective to maintain.Maintainability requirements may cover diverse levels of documentation, such as system documentation, as well as test documentation, e.g. which test cases and test plans will accompany the system.

B) WATERFALL MODEL

It is also referred to as a linear-sequential life cycle model.The Waterfall model is the earliest SDLC approach that was used for software development.The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.



C) UNIT TESTING

Unit testing, a testing technique using which individual modules are tested to determine if there are any issues by the developer himself. It is concerned with functional correctness of the standalone modules.The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

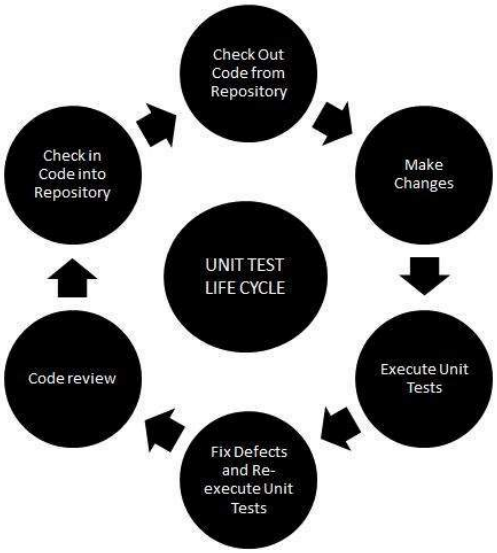
Advantages:

Reduces Defects in the Newly developed features or reduces bugs when changing the existing functionality.

Reduces Cost of Testing as defects are captured in very early phase.

Improves design and allows better refactoring of code.

Unit Tests, when integrated with build gives the quality of the build as well.



Unit Testing Techniques:

Black Box Testing - Using which the user interface, input and output are tested.

White Box Testing - used to test each one of those functions behaviour is tested.

Gray Box Testing - Used to execute tests, risks and assessment methods.