Varun Khadayate A016 07-10-2020

S-E

Assignment-V

Q1. - The main advantage of distributed version control system is that we can work in the network sharing the file in the very manner operating and commit it. And there is no centralized server so the failure of the central network will not impact this.

- The disadvantage of the distributed version control system is it is possible when 2 users update the same field at a time this will cause a problem also the test out is slower as compared to centralized system.

Q2. A critical system is a system that must be highly reliable and retain the reliability as they evolve without incurring prohibitive cost.
There are 3 types of critical system namely: Safety, Mission, Business.

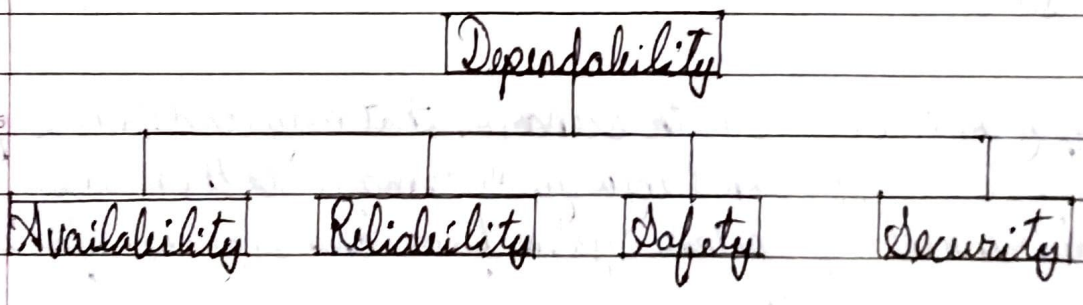
1. Safety Critical: Deals with scenarios that may lead to loss of life, serious personal injury or damage to the natural environment. Eg: Control system for a chemical manufacturing plant, aircraft etc.

2. Mission Critical: Made to avoid the inability to complete the overall system, project objectives, or one of the goals for which the system was designed. Eg: Navigation system of a spacecraft spacecraft etc.

3. Business Critical: Programmed to avoid significant tangible or intangible economic ~~to~~ cost. Eg: Customer accounting system in banks, stock trading System.

Q3. The 6 reasons why dependability in critical system are important are:
- Users may not use the system if they don't trust it.
- System failure may lead to a loss of business.
- An undependable system may lose or damage valuable data.
- An undependable system may damage its external environment.
- The reputation of the company that produced the system may be damaged hence affecting other systems.
- The system may be in breach of laws on consumer protection & the fitness of goods for purpose.

Q5.



1. Availability: The availability of a system is the probability that it will be up and running and able to deliver useful services to any given time.

2. Reliability: The reliability of a system is the probability, over a given period, that the system will correctly deliver services as expected by the user.

3. Safety: The safety of a system is a judgement of how likely it is that the system will cause damage to people or its environment.

4. Security: The security of a system is a judgement of how likely the system can resist accidental or deliberate intrusion.

Q6. 1. Availability, as the failure of availability affects a large number of people, the reputation of the supplier, and hence its current & future income.

2. Safety, as safety related failures can cause harm to the patient.

3. Reliability, as mission failure could result from failure of the system to perform to specification.

4. Security, because of potential losses to users.

Q7. - Microwave
- Electric Drill
- Lawn Mower
- Vacuum Cleaner
- Blender
- Central Heating Furnace.

Q8. Ensuring system reliability does not necessarily lead to system safety as reliability is concerned with meeting the system specification whereas safety is concerned with excluding the possibility of dangerous behaviour. If the specification does not explicitly exclude dangerous behaviour then a system can be reliable but unsafe.

Q9. A possible hazard is the delivery of too much radiation to a patient. This can arise because of a system failure where a dose greater than the specified dose is delivered or an operator failure where a dose greater to be delivered is wrongly input.
Software features that may be included to guard against system failure are the delivery of radiation in increments with an operator display showing the dose delivered and the requirement that the operator confirms the delivery of the next increment. To reduce the probability of operator error, there could be a feature that requires confirmation of the dose to be delivered to that patient.

Q10. • Anti-lock braking system: This is a safety-critical system so requires a lot of up-front analysis before implementation. It certainly needs a plan-driven approach to development with the requirements carefully analysed. A waterfall model is therefore the most appropriate approach to use, perhaps with formal transformation between the different development stages.

• Virtual reality system: This is a system where the requirements will ~~be~~ change and there will be an extensive user interface component. Incremental development with, perhaps, some UI prototyping is the most appropriate model. An agile process may be used.

• University accounting system: This is a system whose requirements are fairly well-known & which will be used in an environment in conjuction with lots of other system such as a research grant management system. Therefore, a reuse-based approach is likely to be appropriate for this.

• Interactive travel planning system: System with a complex user interface but which must be stable & reliable. An incremental development approach is the ~~work~~ most appropriate as the system requirements will change as real user experience with system is gained.

Q11. The specification of evolutionary development projects are often abstract, and as the project continues, the development and validation portions of software engineering overlap one another. This usually results in the system being poorly constructed due to a good initial specification, and on large projects makes it more difficult to integrate new systems into evolutionary design. Lastly, the documentation for such projects is often lacking, as the design are constantly rebuilt to the customers specification.

Q12. The spiral model is much like waterfall model, in that there are well-defined stages, but different in that once an initial sequence is complete, the process starts over, correcting problems & expanding ideas to better ~~ideas~~ suit the customer's needs. Also, as an iteration of the spiral may be short, a prototype can be produced that the customer can see & work with, to guide the engineers to construct more accurately what they seek.

Q13. The Rational Unified Process recognizes that a single process model presents only a single view, so the RUP incorporates 3 perspectives, 2 of them being dynamic, that describes the ~~activity~~ phases of the model as time progresses, and static that describes the activities that are enacted. The strength in using both is that phases of the development process are not tied to any one specific workflow, so the entire process can be understood.

Q4. Computer Aided Software Engineering [CASE] is the implementation of computer facilitated tools & methods in software development. It is used to ensure a high-quality & defect free software.

Categories

1. Front End Tools: Also called as Upper Case Tools & are used during preliminary investigation, project planning, analysis & design.

2. Back End Tools: Also called as Lower Case Tools & are used during development, implementation evaluation & maintainance.

3. Cross Life Cycle Tools: Used for support on-going activities such as project management and creating documentation, which occur across multiple phases of system development.

Types

1. Diagramming Tools: Helps in diagramatic & graphical representation of data & system process. It represents system elements, control flow & data flow among different software components & system structures in pictorial form.

2. Computer Display & Report Generators: Helps in understanding the data elements & its relationship involved.

3. Analysis Tools: Focuses on inconsistent, incorrect specification involved in the diagram and data flow. It helps in collecting requirements, automatically checks for any irregularity, data redundancies.

4. Central Repository: Provides a single point of storage for data diagrams, reports and documents related to project management.

5. Documentation Generati

5. Documentation Generators: Helps in generating user and technical documentation as per standards. It creates documents for technical & end user.

6. Code Generators: Aids in the auto generation of code, including definitions, with help of design, documents & diagram.

Q14. A software process model is a simplified description of software process that represents one's view of that process. Process models may include activities that are part of software process, software products & roles of people involved in software engineering.
Some of process model are:

1. Workflow Model: Shows the sequence of activities in the process along with their inputs, outputs and dependencies. The activities in this model represents human actions.

2. Dataflow Model: Represents the process as a set of activities each of which carries out some data transformation. It shows how the input to the process, such as specification, is transformed to an output, such as design.

3. Role Relation Model: Represents the roles of people involved in software process and activities for which they are responsible.

Most software process models are based on one of these general models of software development:

1. Waterfall Approach: This takes above activities and represents them as seperate processes phases such as requirements specification, software design, implementation & so on.

2. Iterative Approach: This approach interleaves the activities of specification, development and validation. An initial system is rapidly development from very abstract specification. This is then refined with customer input to produce a system that satisfies the customer needs.

3. Component Based Approach: This technique assumes that parts of system already exist. The system development process focuses on integrating these parts than developing them from scratch.

Q17. The reasons are:-

1. The use of more ~~costly~~ development expensive techniques and hardware that are required to achieve the higher levels of dependability.
2. The increased testing & system validation that is required to convince the system client that required levels of dependability have been achieved.

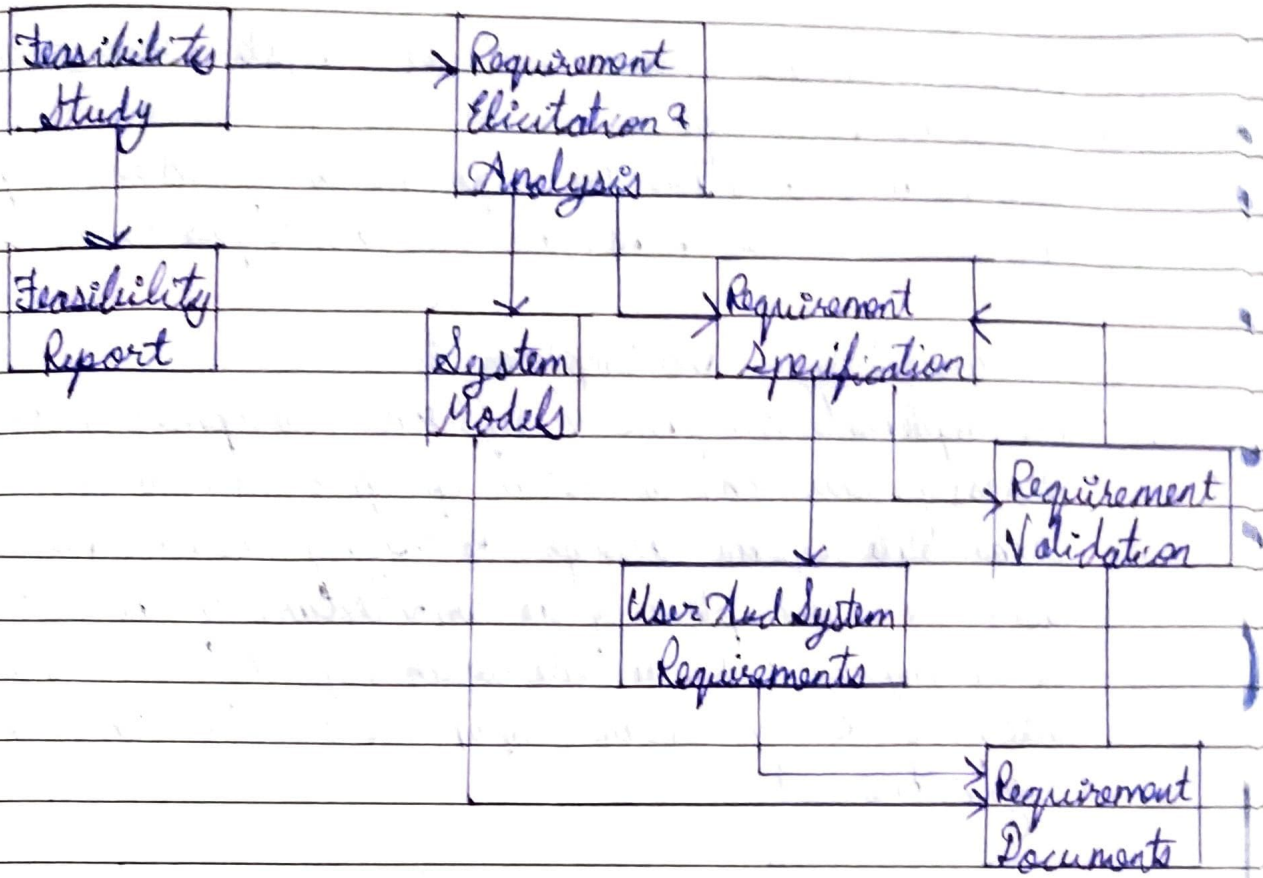
Q16. Process Activities are the activities that belong to a particular process. It defines the smallest measurable amount of work to be performed to convert some portion of process activities inputs into desired outputs. The 4 basic process activities of specification, development, validation, and evolution are organised differently in different development processes. In the waterfall model, they are organised in sequence whereas in evolutionary development, they are interleaved. How these activities are carried out depend on the type of software, people and organizational structures involved.

The 4 process activities are:-

- Software Specification
- Software Design and Implementation
- Software Validation
- Software Evolution

1. Software Specification

A software requirement is defined as a condition to which a system must comply. Software specification or requirement management is the process of understanding and defining what functional and non-functional requirements are required for the system and identifying the constraints on the system operation and development. The requirement engineering process results in the production of a software requirement document that is the specification for the system.



There are 4 main phases of requirement engineering process:-

1. Feasibility Study
In this study an estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. It considers whether the proposed system will be cost effective from a business point of view and ~~whether~~ whether it can be developed within existing budgetary constraints

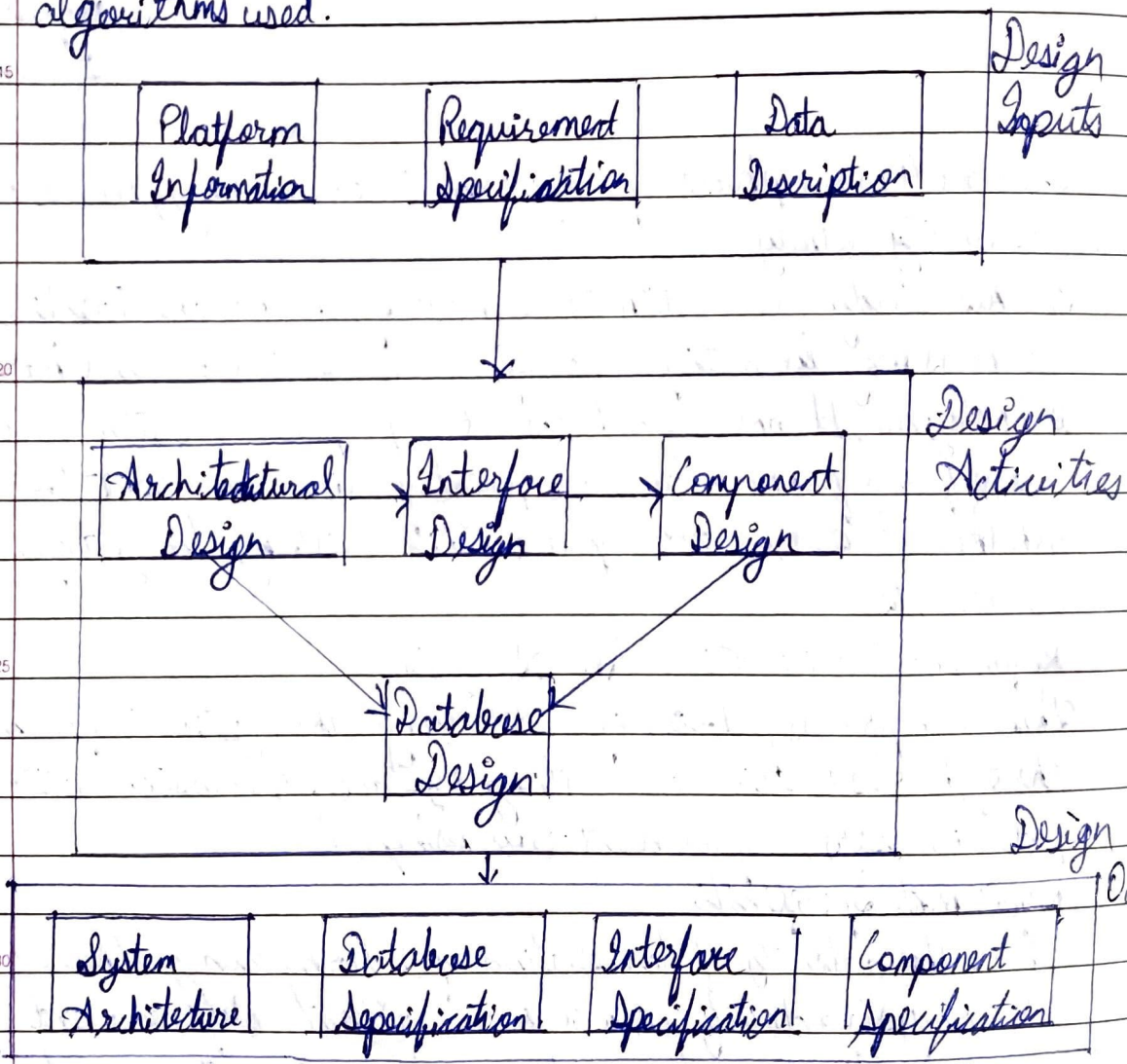
2. Requirement Elicitation And Analysis
This is the process of ~~determining~~ deriving the system requirements through observation of existing systems, discussion with potential users, requirement workshops, etc.

3. Requirement Specification
This is the activity of translation of information gathered during the analysis activity into a document that defines a set of requirements. Two ~~types~~ types of requirement may be

included in these documents: User & System requirements.

4. Requirement Validation
It is determined whether the requirements defined are complete. This activity also checks the requirements for consistency.

2. Software Design and Implementation
The implementation phase of software development is the process of converting a system specification into an executable system through the design of system. A software design is a description of the architecture of the software to be implemented, the data which is part of the system, the interfaces between system components and sometimes, the algorithms used.



The design process activities are the following:

1. Architectural Design
The sub-system of a system and their relationships are identified based on their main functional requirements of software.

2. Abstract Specification
For each subsystem, an abstract specification of its services and the constraints under which it must operate is defined.

3. Interface Design
Interface allow the subsystem services to be used by other sub-systems. The representation of interface should be hidden. In this activity, the interface is designed and documented for each subsystem. The specification of interface must be unambiguous.

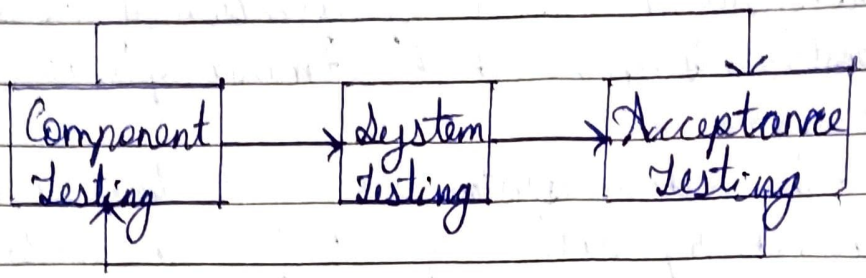
4. Component Design
Services are allocated to components and the interfaces of these components are designed.

5. Data Structure Design
The data~~base~~ structures used in the system implementation are designed in detail and specified.

6. Algorithm Design
The algorithms used to provide services are designed in detail and specified.

3. Software Validation
Software Validation or more generally, verification and validation is intended to show that a system conforms its specification and that the system meets the expectation of the customer buying the system. It involves checking the process at every stage of the software process. The majority of validation costs are incurred after implementation on when the operation of system is tested. The software is

in the usual 3-stages testing process. The system components, the integrated system and finally the entire system are tested. Component defects are generally discovered early in the process and the interface problems during the system integration.



The stages in testing process are:-

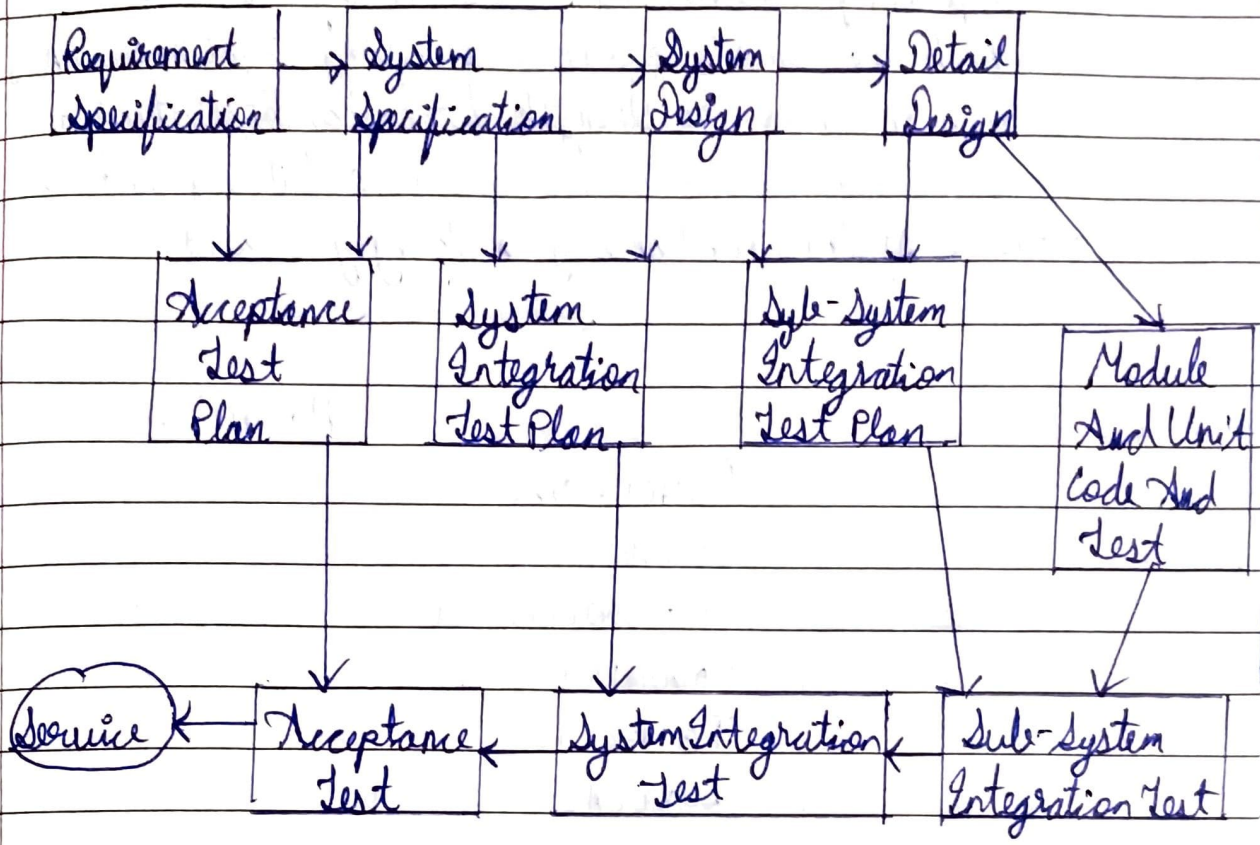
1. Component Testing.
Individual component are tested to ensure that they operate correctly. Each component is ~~used~~ tested independently, without other system component.

2. System Testing
The components are integrated to make up the system. This testing process is concerned with finding errors that result from iteration between components and component interface problems. It is also concerned with validation that the system meets its functional and non-functional requirements.

3. Acceptance Testing
It is considered a functional testing of system. The system is tested with data supplied by the system customer.

Usually component development & testing are interleaved. Programmers make their own test data and test the code as it is developed. However in many process model, such as in V-model, Test-Driven Development etc. The design of

test cases starts before the implementation phase of development. If an incremental approach to development is used, each increment should be tested as it is developed, with these test based on the requirements for that increment.



4. Software Evolution

Software evolution, specifically software maintainance, is the term used in software engineering to refer to the process of developing software initially, then repeatedly updating it for various reason. The aim of software evolution would be to implement the possible major changes to the system. The existing larger system is never complete and continuous to evolve. As it evolves, the complexity will grow. The main objective of software evolution and ensuring the reliability and flexibility of the system. The cost of maintainance are often several times the initial development cost of software

Q15. Iterative development is a way of breaking down the software development of a large application into smaller chunks. In iterative development, feature mode is designed, developed & tested in repeated cycles. With each iteration, additional features can be designed, developed and tested until there is a fully functional software application ready to be developed the customer.