SE 2017-2021 Question Papers Unit-Wise Solutions

Unit 1

Scrum

1. ### Discuss scrum roles with their responsibilities

   In the scrum approach, there are several roles that are important to its success:

   - **Product Owner:** responsible for defining the features and priorities of the product
   - **Scrum Master:** responsible for facilitating the team and ensuring that the scrum process is being followed
   - **Scrum Members** (Development team): responsible for completing the work necessary to build the product

2. ### Discuss scrum events and their relevance

   Sprint: the time-boxed iteration of work in the scrum approach, typically lasting two to four weeks. The purpose of the sprint is to complete a specific set of work, known as the sprint goal, and to deliver a potentially shippable product increment. By breaking the work down into smaller, time-boxed iterations, the team can stay focused and adapt to changing requirements or priorities.

   Sprint Planning: a meeting held at the beginning of each sprint, where the team discusses and plans the work to be completed in the upcoming sprint. The team, along with the Product Owner, will review the product backlog and identify the items that need to be worked on in the sprint. They will then discuss and estimate the work, and create a sprint goal that defines the work to be completed in the sprint.

   Daily Scrum: a short meeting held every day, where each team member provides a brief update on their progress and any obstacles they are facing. The purpose of the daily scrum is to keep the team focused and on track, and to identify and address any issues that may be blocking progress.

   Sprint Review: a meeting held at the end of each sprint, where the team demos the work completed in the sprint to stakeholders. The team will present the product increment that was created during the sprint, and demonstrate the new features and functionality. This is an opportunity for the stakeholders to provide feedback and direction for the next sprint.

   Sprint Retrospective: a meeting held at the end of each sprint, where the team reflects on the sprint and identifies ways to improve in the future. The team will discuss what went well during the sprint, what could be improved, and what actions can be taken to address any issues. This event is an important part of continuous improvement in the scrum process.

3. ## Discuss sprints and their relevance to make scrum approach agile

   - A sprint is a time-boxed iteration of work in the scrum approach, typically lasting two to four weeks
   - **Purpose:** complete a specific set of work (the sprint goal) and deliver a potentially shippable product increment
   - **Benefits:** allows the team to stay focused and adapt to changing requirements or priorities
   - **Sprint planning meeting:** review the product backlog and identify work for the sprint; discuss and estimate the work; create a sprint goal
   - **During the sprint:** team focuses on completing the work necessary to achieve the sprint goal; hold daily stand-up meetings; work collaboratively to overcome challenges
   - **At the end of the sprint:** demo the product increment to stakeholders and gather feedback for the next sprint
   - Overall, sprints are a critical part of the scrum approach, as they **provide a structured way for the team to deliver working software in a predictable and repeatable manner**. This helps to make the scrum approach agile and effective.

4. ## Discuss scrum agile approach in terms of its activities with advantages and disadvantages

   The scrum approach to agile software development is a framework that emphasizes collaboration, accountability, and adaptability:

   - Based on the principles of the agile manifesto
   - Development team works in short sprints to deliver small, incremental improvements to the product
   - Allows the team to be flexible and responsive to changing customer needs and requirements

5. ## What is agile manifesto? How does the scrum model adhere to the agile manifesto?

   The agile manifesto is a set of guiding principles for agile software development:

   - Created in 2001 by a group of software developers to provide a common framework for agile teams
   - Values individuals and interactions, working software, customer collaboration, and responding to change OVER processes and tools, comprehensive documentation, contract negotiation, and following a plan
   - The scrum model adheres to the agile manifesto by emphasizing collaboration, adaptability, and delivering working software.

6. **Discuss Project Backlog**
    - A project backlog is a prioritized list of work items, such as features, user stories, or tasks, that need to be completed in order to achieve the project goals by working on the most important items first.
    - It is the source of work for the development team, as they select items from the backlog and complete them during each sprint.
    - It is a dynamic document that is constantly evolving, as new work items are added, priorities change, and work is completed.
    - It is owned by the Product Owner, who is responsible for maintaining it and ensuring that it is up-to-date.
    - The project backlog helps to keep the team focused and on track, as it provides a clear view of the work to be done and the progress being made.

In Agile Approach, state True or False

1. **Individuals and Interactions are considered more important than processes and tools**

   True
2. **Comprehensive documentation is more important than working software**

   False
3. **Responding to changes in the requirements are more important than following a plan**

   False
4. **Products developed are available for customers at the end of the development lifecycle**

   False
5. **Contract negotiations with customers are not as important as customer collaboration**

   True

True or False

1. **Requirements can be changed with any SDLC**

   True
2. **V Model has testing happening in parallel with development**

   True
3. **Pair Programming is a practice which can be used as part of any agile or plan driven lifecycle**

   True

4. **CDLC differs from Plan or Agile Lifecycles in Packing and Distribution Phases**

   True

   (CDLC, or the Continuous Delivery Lifecycle, is a software development methodology that focuses on frequent, small releases of code changes. This differs from traditional plan-driven or agile lifecycle methods, which typically involve larger, less frequent releases.)

5. **While preparing the project plan, we need to plan for documentation activities as well.**

   True

6. **In Scrum model, at the end of every week, a working version of the software should be available.**

   False

7. **An advantage of the CBSE is that components could be used as a black box and hence reduces complexity of the systems**

   False

   It allows developers to reuse existing components in their software, rather than having to build all the components from scratch. This can save time and effort, and can help to ensure that the components are well-tested and reliable. However, the components may still be complex in themselves, and the interactions between the components may add additional complexity.

Fill in the blanks

1. **The model where the existing code is not modified to a great extent while adding new features is _ _ _ _ _ _ _.**

   incremental

2. **The model where the existing code is modified to a great extent while adding new features is _ _ _ _ _ _ _ _ _ .**

   evolutionary

3. ## What is a risk and what are the steps (with one line description each) involved in managing risks?

   A risk is a potential negative outcome or danger that may arise in a situation or project. The steps involved in managing risks are:

   - **Identifying risks:** Identify the potential risks that may arise in a situation or project.
   - **Analyzing risks:** Evaluate the potential impact and likelihood of identified risks.
   - **Prioritizing risks:** Rank the identified risks based on their potential impact and likelihood.
   - **Developing risk response strategies:** Develop strategies to mitigate or avoid the identified risks.
   - **Implementing risk response strategies:** Put the developed strategies into action to reduce or eliminate the identified risks.
   - **Monitoring and reviewing risks:** Continuously monitor and review the identified risks and the effectiveness of the implemented strategies.

4. ## What is CBSE? What are its advantages and disadvantages?

   CBSE (Component-Based Software Engineering) is a software development methodology that focuses on building software systems using reusable components.

   The main advantages of CBSE are:

   - **Reusability:** Components can be reused in different contexts, which can save time and effort in software development.
   - **Modularity:** Components can be designed to be modular and easily understood, which can make the system more maintainable.
   - **Flexibility:** Components can be easily replaced or upgraded, which can improve the adaptability and flexibility of the system.

   The main disadvantages of CBSE are:

   - **Complexity:** The use of components can add complexity to the system, which can make it more difficult to understand and maintain.
   - **Integration:** Components must be carefully integrated to work together, which can be a time-consuming and error-prone process.
   - **Compatibility:** Components must be compatible with each other and the overall system, which can be a challenge in large and complex systems.

5. ## Discuss the steps involved in the Requirements Engineering Process along with what is being achieved in each step

   The goal of requirements engineering is to ensure that the resulting software or product will meet the needs of the end user or customer.

   The requirements engineering process typically involves several steps, including:

   1. **Elicitation:** In this step, the requirements engineer works with stakeholders, such as end users, customers, and subject matter experts, to gather and collect information about the needs and requirements for the software or product. This may involve techniques such as interviews, surveys, focus groups, or workshops.
   2. **Analysis:** In this step, the requirements engineer analyzes the information collected in the elicitation step to identify common themes, patterns, and requirements. This may involve organizing the requirements into categories or using modeling techniques to represent the requirements in a visual or graphical form.
   3. **Documentation:** In this step, the requirements engineer documents the requirements in a clear and concise manner, using a standardized format and notation. This may involve creating a requirements document or specification that outlines the functional, performance, and non-functional requirements for the software or product.
   4. **Validation:** In this step, the requirements engineer works with stakeholders to validate and verify that the requirements accurately reflect the needs and requirements of the end user or customer. This may involve techniques such as user acceptance testing, where the requirements are tested against real-world scenarios to ensure they are accurate and complete.
   5. **Management:** In this step, the requirements engineer manages and maintains the requirements over the course of the software development or product development process. This may involve tracking changes to the requirements, prioritizing requirements, and communicating changes to the development team.

6. Contrast SDLC and PDLC

Software Development Life Cycle (SDLC) and Product Development Life Cycle (PDLC) are both systems that organizations use to plan, develop, and manage the creation of new products or software.

The main difference between the two is that SDLC focuses on the development of software, while PDLC focuses on the development of a product, which can include software but can also include other types of products, such as physical goods or services.

Both SDLC and PDLC typically involve several stages, such as planning, design, development, testing, and deployment. However, the specific stages and their focus may vary depending on the specific approach or methodology being used.

For example, SDLC may include stages such as requirements gathering, design, coding, testing, and maintenance, while PDLC may include stages such as market research, product design, prototype development, testing, and product launch.

Overall, the goal of both SDLC and PDLC is to ensure that new products or software are developed in a structured and organized manner, with the needs and requirements of customers or users in mind.

7. Contrast Waterfall Model, CBSE and Product line in terms of Reusability and Supportability

The **Waterfall model** is a software development model that is based on a linear, sequential approach. In this model, each phase of the software development process must be completed before the next phase can begin, and there is little or no overlap between phases. This model is not designed with reusability or supportability in mind, and as such it is not particularly well-suited for these purposes.

The **CBSE** (Component-Based Software Engineering) approach is a software development method that is based on the reuse of existing software components. In this model, software is built using a set of pre-existing, modular components that can be easily combined to create new applications. CBSE is a good choice for projects that require high levels of reusability and supportability, since it allows for the easy reuse of code and the creation of software that is easy to maintain and modify.

A **product line** is a set of related products that share a common, standardized set of features. In software development, a product line approach is one in which a set of core components and features are developed and then used as the basis for a range of related software products. This approach allows for high levels of reusability and supportability, since the core components can be easily reused and maintained across multiple products.

8. Discuss the V model of SDLC with the different phases, focus on each of these phases and the deliverables or artifacts which you would see at the end of the phase. Discuss how is this similar to a waterfall model and how it is different from a waterfall model.

The V model of the software development life cycle (SDLC) is a graphical representation of the process of developing software. It is similar to the waterfall model in that it is a sequential, linear approach to software development, with each phase building upon the previous one. However, it is different from the waterfall model in that it includes specific,

defined phases for testing and verification, which are integrated with the corresponding phases of development.

The V model consists of the following phases:

1. **Requirements gathering and analysis:** In this phase, the software requirements are gathered and analyzed to determine the scope and functionality of the software.
2. **Design:** In this phase, the software architecture and design are created, including the overall structure and organization of the software, as well as the specific components and modules that will be implemented.
3. **Implementation:** In this phase, the individual components and modules of the software are developed and implemented according to the design specifications.
4. **Integration and testing:** In this phase, the individual components and modules are integrated and tested to ensure that they work together correctly.
5. **Verification and validation:** In this phase, the software is tested to ensure that it meets the specified requirements and works as intended.

At the end of each phase, there are specific deliverables or artifacts that are produced. For example, at the end of the requirements gathering and analysis phase, a requirements specification document is produced. At the end of the design phase, a design specification document is produced. At the end of the implementation phase, the individual components and modules of the software are produced. And at the end of the integration and testing phase, a complete, working version of the software is produced.

The V model is similar to the waterfall model in that it is a sequential, linear approach to software development. However, it is different from the waterfall model in that it includes specific phases for testing and verification, which are integrated with the corresponding phases of development. This allows for a more thorough and comprehensive approach to software development, and helps to ensure that the final product is of high quality and meets the specified requirements.

Case study questions

1. **Assume you are developing an application which will support online booking and selling of movie tickets. Specify any 3 function and 3 non-functional requirements for the same, keeping the properties expected of a requirement.**

   Functional requirements:

   1. The application should allow users to search for movies by title, genre, and location.
   2. The application should allow users to select a specific showing of a movie and choose their seats.
   3. The application should allow users to enter payment information and complete the purchase of their movie tickets.

   Non-functional requirements:

   1. The application should have a user-friendly interface and be easy to navigate.
   2. The application should be secure and protect user information, such as payment details.
   3. The application should be able to handle high volumes of traffic and concurrent users without crashing or experiencing significant delays.

2. **Assume you are the project manager of two projects with the following characteristics:**

    - **Project 1: A real-time system which is complex but whose requirements can be relatively easily identified and are stable**
    - **Project 2: A web-site for a local clinic. Requirements are vague and are likely to change in the future.**

    Consider the following software development approaches/models: waterfall, incremental, evolutionary prototyping, throw-away prototyping and component-based development.

    (i) Which of the above models will you choose for each of your projects?

    (ii) Justify your choice for each project

    (i) **For Project 1, I would choose the waterfall model.** This model is well-suited for a complex real-time system with stable requirements because it allows for a structured and linear development process, with each phase of the project building on the previous one. This will help ensure that the final product meets the requirements and is delivered on time.

    For Project 2, I would choose the incremental model. This model is well-suited for a project with vague and changing requirements because it allows for a more flexible development process. Instead of trying to identify all requirements upfront, the incremental model allows for the gradual development and refinement of the system, with each iteration building on the previous one. This will help ensure that the final product meets the evolving needs of the clinic.

    (ii) The waterfall model is a good fit for Project 1 because it allows for a structured and linear development process that is well-suited for a complex real-time system with stable requirements. The incremental model is a good fit for Project 2 because it allows for a more flexible development process that can accommodate the vague and changing requirements of a web-site for a local clinic. Both models will help ensure that the final products meet the specific needs of each project.

3. **What is the problem with the following requirement statement? That is, what quality attributes the statement does not possess? How can it be improved?**

    "**SR-4: The pages of the application will load in acceptable amount of time.**"

    The problem with the requirement statement is that it is not specific or measurable. The term "acceptable amount of time" is vague and subjective, and it is not clear what would constitute an acceptable amount of time for the pages to load. In order to improve the requirement statement, it should be made more specific and measurable.

    For example, the requirement could be revised to state something like "SR-4: The pages of the application will load within 3 seconds or less." This revised statement is more specific and measurable, and it clearly defines what would constitute an acceptable amount of time for the pages to load.

4. **What is a software lifecycle model? What are the common characteristics of each phase or activity in a software lifecycle model?**

   A software lifecycle model is a framework that defines the stages and activities involved in the development of a software system. Each phase or activity in a software lifecycle model has certain characteristics that are common across different models. Some of these common characteristics are:

   1. Each phase or activity has a **specific goal or objective** that it aims to achieve.
   2. Each phase or activity has **well-defined inputs, outputs, and deliverables**.
   3. Each phase or activity has **specific activities, tasks, or steps** that need to be performed in order to achieve its goal.
   4. The **progress** and completion of each phase or activity is typically **reviewed and verified by stakeholders**.
   5. The **output** of each phase or activity **serves as input for the next phase** or activity in the software lifecycle model.

   Overall, the common characteristics of each phase or activity in a software lifecycle model help to ensure that the development of the software system is structured, organized, and efficient.

Glossary

CBSE - Component Based Software Engineering

PDLC - Product Development Lifecycle

SDLC - Software Development Lifecycle

CDLC - Component Development Lifecycle

Unit 2

Subjective

1. **Describe in 2-3 sentences, any 4 principals/techniques that influence a good design**

   Four principles that influence a good design are:

   • **Balance:** Achieving a sense of equilibrium among the elements in a design.

   • **Contrast:** Creating visual differences to draw attention to important elements.

   • **Repetition:** Using the same elements throughout a design to create consistency.

   • **Hierarchy:** Creating a logical order of importance to emphasize key elements.

2. **Describe in 1-2 lines any six approaches that can be taken to decompose a larger system into subsystems or components**

   Six approaches that can be taken to decompose a larger system into subsystems or components are:

   • **Top-down decomposition:** Breaking a system into the highest-level components, and then breaking each of these components into further sub-components.

   • **Bottom-up decomposition:** Breaking the system into its smallest components and then gradually combining them to form larger components.

   • **Goal-oriented decomposition:** Breaking the system into components based on the goals of the system.

   • **Feature-driven decomposition:** Breaking the system into components based on the features of the system.

   • **Domain-driven decomposition:** Breaking the system into components based on the domain of the system.

   • **Object-oriented decomposition:** Breaking the system into components based on objects.

3. **Discuss any two properties which you would expect from a software requirement with an example to illustrate the same**

   Two properties which you would expect from a software requirement are:

   • **Understandability:** The requirement should be clearly understood and unambiguous.

   • **Testability:** The requirement should be testable and measurable.

   Example: A software requirement to “improve the user experience” is not testable since there is no way to measure the level of improvement.

4. **Discuss an approach through which you could choose a lifecycle for development of a product. Use two examples on how a lifecycle was chosen by two different companies**

   An approach for choosing a lifecycle for development of a product could include:

   • Understanding the market and customer needs.

   • Creating a product vision and breaking it down into achievable goals.

   • Determining the resources needed for the project and the timeline for completion.

   • Assigning roles and responsibilities for the team.

   Examples:

   • **Company A:** Company A chose an Agile lifecycle, which focuses on short sprints and frequent customer feedback.

   • **Company B:** Company B chose a Waterfall lifecycle, which follows a linear process from planning to development and testing.

5. **Enumerate and briefly discuss in 1-2 sentences, any 5 considerations which influence the software design characteristics like simplicity and maintenance**

   Five considerations which influence the software design characteristics like simplicity and maintenance are:

   • **Usability:** The design should make it easy for users to accomplish their tasks.

   • **Performance:** The design should make efficient use of resources.

   • **Security:** The design should protect the system from security threats.

   • **Scalability:** The design should allow for future growth.

   • **Maintainability:** The design should be easily maintained and updated.

6. **Discuss briefly any 4 factors (drivers) that influence Software Architecture**

   Four factors (drivers) that influence Software Architecture are:

   • **Goals:** The architecture should meet the business and technical goals of the system.

   • **Technology:** The architecture should take full advantage of the available technology.

   • **Platform:** The architecture should be tailored to the target platform.

   • **Budget:** The architecture should be within the budget constraints of the project.

7. **In the context of project planning and control, discuss the following**

1. **Upstream and Downstream dependencies**
2. **Risks in terms of identification, trigger and mitigation**

1. **Upstream and Downstream dependencies:** Upstream dependencies are tasks or activities that must be completed before the current task can begin, while downstream dependencies are tasks or activities that must be completed after the current task is complete.
2. **Risks in terms of identification, trigger and mitigation:** Risk identification is the process of finding, recognizing, and describing risks. Risk triggers are events that can cause risks to occur. Risk mitigation is the process of reducing the impact of risks.

8. **Discuss one Architectural pattern which you could use in building the software for the car service center, in terms of generic description of the pattern, the benefit of using the pattern and potential issue/liability of using the pattern.**

   The Model-View-Controller (MVC) architectural pattern is a commonly used pattern for software development. It divides an application into three distinct parts: the model, the view, and the controller. The model contains the data and logic of the application, the view is the user's interface with the application, and the controller is responsible for handling user input and updating the model and view. The benefits of using the MVC pattern include better code organization and separation of concerns, improved testing and debugging, and improved scalability. Potential issues of using the MVC pattern include difficulty in understanding the pattern, and potential performance issues due to the multiple layers of the pattern.

9. **Consider you are building a plan for a project, where you have broken down the project requirements into a set of executable tasks and have estimated the amount of effort needed to execute that. Discuss any 6 points which need to be considered while building a schedule for the project**

   Six points to consider when building a schedule for a project:

   • **Resource availability:** Identifying the availability of resources such as people, materials, and equipment.

   • **Sequence of activities:** Determining the order of activities and dependencies.

   • **Estimating duration:** Estimating the time needed to complete each activity.

   • **Risk management:** Identifying and mitigating risks.

   • **Budget and cost:** Estimating the cost of the project.

   • **Scheduling software:** Identifying and using scheduling software to create the project schedule.

Case Study

1. **Consider your company has a software product which has software components deployed across different servers, and when a problem occurs, support engineers need to trouble shoot using the information logged by the software components. Some of the characteristics of the environment are that Log sizes are large, and these logs are generated at small intervals of time, and the interest in the logs is for particular events over specific time period in the logs. You are expected to provide an Architectural solution to support quick and effective trouble shooting.**

   There are potentially different architectural approaches/solutions to solve the above problem, where each of the solutions will have different trade-offs in terms of non- functional requirements. Illustrate this by providing 3 solutions for the above scenario highlighting what non-functionalities and the trade-off therein.

   Solutions:

   • **Solution 1:** Store the logs in a centralized location and use a search tool to search for particular events over a specific time period. This would be the simplest solution but may cause performance issues.

   • **Solution 2:** Store the logs on the local servers and use a distributed storage system to aggregate the logs for analysis. This would improve performance but may require additional architecture for distributed storage.

   • **Solution 3:** Store the logs on the local servers and use a distributed computation system to aggregate the logs for analysis. This would improve performance and scalability but would require additional architecture for distributed computation.

2. **An OTT video service website, CutFlix, needs to have the following features.**
   - **There are two types of videos, a movie or an episode of a series. The movie has details like name, release year, producer, description, director and duration.**
   - **The movie also has associated actors. The platform should display the list of actors and the roles played by them, which may include multiple roles played by the same actor.**
   - **An episode belongs to the season of a series. It has a name, release date, producer, director, duration and description.**

- **The season will have details like number of episodes, release year and description. The series has name, description, genres and maturity rating.**
- **Each episode also has associated actors. The platform should display the list of actors and the roles played by them, which may include multiple roles played by the same actor.**
- **A registered user can create multiple profiles. A profile has a list of videos viewed. On deleting the user account, all associated profiles should also be deleted.**
- **A user can add the video to their profile's Watchlist. S/He could remove the video from Watchlist anytime.**
- **The video currently being watched by the user appears in the profile's "Continue watching" list. The user can remove the video from this list at any point of time. The video is automatically removed from the list once the user completes watching the video.**

1. **For the above case study, prepare the class diagram by identifying the classes, their properties, associations, association classes and cardinality**
2. **For the video object described above, prepare the state diagram identifying the states, transactions, events and activities**

Class Diagram

• **Video:** name, release year, producer, description, director, duration, actors

• **Movie:** name, release year, producer, description, director, duration, actors

• **Episode:** name, release date, producer, director, duration, description, season

• **Season:** number of episodes, release year, description

• **Series:** name, description, genres, maturity rating

• **User:** profiles, watchlist, continue watching list

• **Profile:** list of videos viewed

• **Actor:** name, roles

• Associations:

• **Movie-Actor**: many-to-many

• **Episode-Actor**: many-to-many

• **User-Profile**: one-to-many

• **Profile-Video**: many-to-many

• **User-Watchlist**: one-to-many

• **User-Continue Watching**: one-to-many

State Diagram

• **States:** Start, Playing, Paused, Completed, Removed from Watchlist

• **Transactions:** Start Video, Pause Video, Resume Video, Complete Video, Remove Video from Watchlist

• **Events:** Play Video, Pause Video, Resume Video, Complete Video, Remove Video from Watchlist

• **Activities:** Start Video, Pause Video, Resume Video, Complete Video, Remove Video from Watchlist

3. **A Railway reservation system has the following features. It supports a Web GUI which allows Users to query the system for trains between a starting point to a destination based on i) the time of starting and ending ii) accommodation type iii) type of ticket iv) cost. The system also allows registration of users, and when a registered user queries as above, the system allows the user to select an accommodation on the train based on the criteria specified and pay for the ticket through a payment gateway. The system also creates an itinery for the booking and passes back a booking Id which can be used for changes and cancellation.**

    1. **Making assumptions as necessary, build a high-level block level solution for the system.**

    2. **Identify the Upstream and Downstream modules in the above block diagram**

    3. **High-Level Block Level Solution:**

        • User Input Module

        • Train Query Module

        • User Registration Module

        • Payment Module

        • Booking Itinerary Module

        • Booking ID Module

    4. **Upstream Modules:** User Input Module, User Registration Module, Payment Module

Downstream Modules: Train Query Module, Booking Itinerary Module, Booking ID Module

4. **What are Risks, Mitigation Plan and Triggers for Risk Mitigation? Identify one risk for the above project, its mitigation plan and the trigger.**

   Risk: Payment gateway integration failure

   Mitigation Plan: Offer alternative payment processing options such as manual payment processing, or a different payment gateway.

   Trigger: Receive a notification of a payment gateway integration failure.

5. **Discuss all the steps necessary for building a Schedule. Use the above as the example for showing the same.**

   Steps Necessary for Building a Schedule:

   • Gather the requirements of the project, such as the features and timeline.

   • Identify the tasks that need to be completed to fulfill the requirements.

   • Estimate how long each task will take to complete.

   • Create a timeline for completing the tasks.

   • Identify resources needed for completing the tasks.

   • Create a budget for the project.

   • Monitor and adjust the timeline and budget as needed.

6. **Consider yourself to be responsible for development of the software which manages workflow in a car service center. Making assumptions as necessary**

   1. **Discuss the steps involved (using Software Engineering principles) in coming up with a set of requirements which can be used for the development of the system**

   2. **In case a couple of new requirements needed to added into the system, discuss on the change management process which you would use.**

   3. Steps involved in coming up with a set of requirements for the software:

      • **Brainstorming:** Identify the user needs and requirements.

      • **Research:** Gather information on the existing system, similar systems, and the current technology.

      • **Documentation:** Organize the identified user needs and requirements into a formal document.

      • **Validation:** Verify the accuracy and completeness of the requirements document.

   4. Change management process:

      • **Analyze the change request:** Evaluate the proposed changes to determine their impact on the system.

• **Discuss and prioritize the changes:** Consider the priority, cost, and timeline of the change requests.

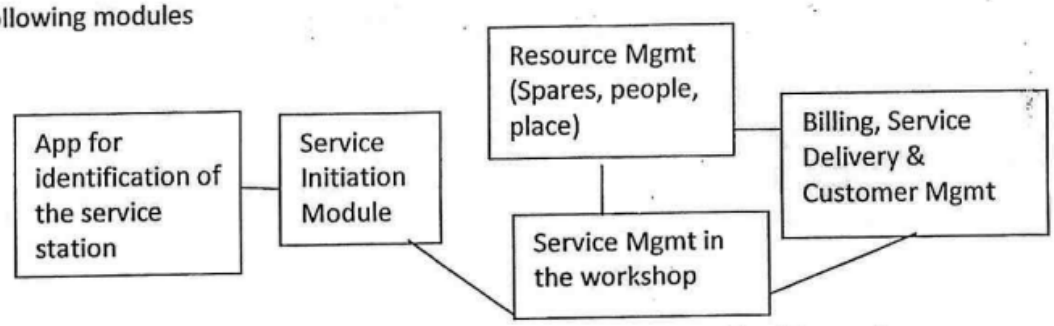
• **Develop a plan for implementing the changes:** Create a plan for implementing the changes in an efficient and effective manner.

• **Test the changes:** Test the changes to ensure they meet the desired requirements.

• **Implement the changes:** Incorporate the changes into the system.

• **Monitor and evaluate the changes:** Monitor the changes to ensure they are having the desired effect.

7. Assume the above software for the car service-center, at a high level, needs to have the following modules



and if you as a person accountable for building this system, and had 5 people resources, and assuming all of the modules have very varied complexity, build a schedule for project, following formal SE planning process activities.

The schedule of the project for the software for the car service center would include the following:

App identification of service station: This module would include identifying the service station based on the location, availability of service technicians, and other relevant details. This module would take approximately 1 week to develop.

Service initiation module: This module would allow customers to initiate a service request and book an appointment. It would include features such as selecting the type of service, scheduling the appointment, and providing customer details. This module would take approximately 2 weeks to develop.

Resource management: This module would allow the service center to manage its resources, such as service technicians, tools, and equipment. It would include features such as scheduling the tasks, allocating resources, and tracking the progress. This module would take approximately 3 weeks to develop.

Service management in the workshop: This module would allow the service technicians to manage the service tasks in the workshop. It would include features such as tracking the progress, updating the status, and providing feedback to the customers. This module would take approximately 2 weeks to develop.

Billing: This module would allow the service center to generate invoices and manage the payments. It would include features such as generating invoices, calculating the charges, and providing payment options. This module would take approximately 1 week to develop.

Overall, the project schedule for the software for the car service center would be approximately 9 weeks.

True or False

1. Cost of repair of a problem found in design is higher to fix than a problem found during implementation
2. Feasibility study is to validate that the product planned for development is good for development
3. Downstream dependencies are those which you are those which you are dependent on
4. Sizability is a non-functional requirement of a product
5. Delphi estimation technique can be effectively applied to estimate an activity, when none of the team members have any experience in the area of activity

Answers

1. False
2. True
3. False
4. False
5. True

MCQs

1. Divide and Conquer is
   A. Architecture Strategy
   B. Implementation Strategy
   C. Testing Strategy
   D. None of the above
2. Which of the following design methodologies first specifies the individual base elements of the system in great detail?
   A. Top down design
   B. Bottom up design
   C. Component-based design
   D. Pattern-oriented design
3. A design that can be can be modified easily to work with other software components during integration is highly
   A. Portable
   B. Profitable
   C. Interoperable
   D. Usable
4. In terms of Cohesion and Coupling
   A. Good to have High Cohesion and High Coupling
   B. Good to have Low Cohesion and High Coupling
   C. Good to have Low Cohesion and Low Coupling
   D. Good to have High Cohesion and Low Coupling
5. Architecture manifests early set of design decisions in terms of
   A. not being predominantly focused towards functional requirement
   B. not setting constraints on implementation
   C. not Inhibiting or enabling quality attributes

D. not driving an organizational structure

6. Which of the following does not belong to the Requirement Analysis Process?
   a. Pareto Analysis
   b. Fish Bone Analysis
   c. De Bono Technique
   d. MOSCOW

7. These are Models used in Requirements Engineering
   a. Prose
   b. ER Diagrams
   c. FSMs
   d. All of the Above
   USN

8. A Product Lifecycle has the following stages

   1.Introduction 2. Growth 3. Maturity 4. Discontinuance 5. Obsolescence

   The order of occurrence of the above stages are
   a. 1,2,3,4,5
   b. 1,2,3,5,4
   c. 1,2,4,3,5
   d. None of the above

9. The following defines the characteristics of a Product Line
   a. Pro-active Approach to Reuse of Software
   b. Bottoms up approach of product development
   c. Product family with many commonalities and few differences
   d. Plan driven approach to product development

10. A design that can be modified easily to run on a variety of hardware and software environments is highly
    a) portable
    b) interoperable
    c) profitable
    d) usable

11. The main difference between Verification and Validation can be explained by the following statement(s) in the order given below:
    a) there is no clear difference - they are essentially the same
    b) verification is for looking at are we building the product right while validation is for looking at whether are we building the right product.
    c) verification involves testing while validation involves process related activities
    d) verification takes place during the testing phase; validation takes place during requirements phase

12. Which of the following software engineering activities typically produce the highest volume of configuration items that need to be managed in a software project?
    a) Software requirement analysis
    b) Software design
    c) Software construction
    d) Software engineering management

13. In software construction, "the discipline of removing unwanted code for easier maintenance and deployment" is known as:
    a) Literate programming
    b) Static analysis
    c) Refactoring
    d) Debugging
14. Which of the following design methodologies first specifies the individual base elements of the system in great detail?
    a) Top down design
    b) Bottom up design
    c) Component-based design
    d) Pattern-oriented design
15. Projects are characterized by
    a) Clarity of Requirements
    b) Clarity of certainty of resources
    c) Clarity of the process to be followed
    d) Clarity of the User environment

Answers:

1. B. Implementation Strategy
2. B. Bottom up design
3. C. Interoperable
4. D. Good to have High Cohesion and Low Coupling
5. C. Not Inhibiting or enabling quality attributes
6. C. De Bono Technique
7. D. All of the Above
8. A. 1,2,3,4,5
9. C. Product family with many commonalities and few differences
10. A. Portable
11. B. Verification is for looking at are we building the product right while validation is for looking at whether are we building the right product.
12. B. Software design
13. C. Refactoring
14. B. Bottom up design
15. A. Clarity of Requirements

Unit 3

Subjective

1. **What are SCM Directories? Provide one example of an SCM Directory and discuss one its challenges**

   SCM Directories are directories that are used to store and manage source code, configuration files, and other related files. An example of an SCM Directory is a Git repository. One challenge associated with SCM Directories is the need to ensure that access to the files is properly secured and managed.

2. **What are Configuration Items? Provide one example of Configuration Item and discuss one of its challenges**

   Configuration Items (CIs) are the items that are managed throughout the change management process. An example of a Configuration Item is a software application. One challenge associated with Configuration Items is the need to ensure that the Configuration Items are properly identified, documented and tracked, so that the changes to the Configuration Items can be managed accordingly.

3. **What is Baselining? Provide one example of a Baseline and discuss one its challenges**

   Baselining is the process of creating a baseline that identifies the state of a system at a given point in time. An example of a baseline is a snapshot of the current version of a software application. One challenge associated with baselining is that it can be difficult to ensure that all changes to the system are captured in the baseline.

4. **Discuss each of these in the context of Software Implementation**

   1. **A program is for the programmer who reads it, not for the computer that runs it.**

      A program is for the programmer who reads it, not for the computer that runs it: The code should be written in a way that a programmer can easily comprehend it, and re-use or modify it as needed. This will reduce the cost of maintenance.

   2. **Programs have "personalities"**

      Programs have "personalities": A program should have a clear purpose and should be written in a way that expresses the intention of the programmer. This will help reduce the time and effort spent on debugging and maintenance.

3. ### Don't patch bad code - re-write it

   Don't patch bad code - re-write it: Writing patches for bad code can lead to an unmaintainable codebase. If a code needs to be modified, it should be re-written instead of patched. This will ensure that the code is maintainable in the long run.

5. ## In the context of software Implementation, discuss the journey of a Bug. What is its relationship to the cost of maintenance? What would you infer from the same?

   The journey of a bug begins with its identification. This is usually done through testing and debugging. Once a bug is identified, it is then documented, and the steps to reproduce it are documented. This information is used to determine the root cause of the bug, and to identify potential solutions. The bug is then fixed, and the code is tested to ensure that the bug has been resolved. The cost of maintenance is directly related to the number of bugs present in the system, as well as the complexity of the bugs. The more bugs there are, the more time and effort is needed to fix them, which increases the cost of maintenance.

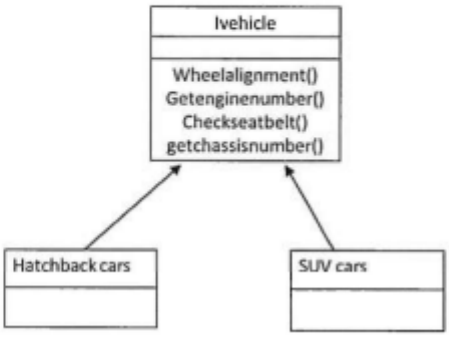
6. ## Discuss each of these terms in the context of Agile Construction

   1. ### Sprint Burndown
   2. ### Team Velocity Metric
   3. ### Throughput

   - **Sprint Burndown:** The sprint burndown is a chart used to track the progress of a sprint. It shows the number of remaining tasks for the sprint and the amount of work completed over time.
   - **Team Velocity Metric:** The team velocity metric is a measure of the amount of work that a team can complete in a given amount of time. It is used to estimate how long it will take to complete a project and to track progress over time.
   - **Throughput:** Throughput is the rate at which tasks are completed in an agile development process. It measures how quickly a team can complete tasks, and is used to measure the efficiency of the team.

7.

Class design given below has functionalities of vehicles. Now, the client wants to add a two wheeler class to the design. If the new two wheeler class implements interface Ivehicle, which design principle is violated? Modify the design in the given class diagram to accommodate the new class.



The design principle that is violated is the Interface Segregation Principle. This principle states that clients should not be forced to depend on methods they do not use. Since the two wheeler class would be required to implement the Checkseatbelt() method, which it cannot use, this principle is violated.

The design can be modified as follows:

Ivehicle -> Wheelalignment() Getenginenumber() getchassisnumber()

IvehicleWithSeatbelts -> Checkseatbelt()

Hatchback cars -> IvehicleWithSeatbelts

SUV cars -> IvehicleWithSeatbelts

Two wheelers -> Ivehicle

8. **Differentiate between Coding standards and guidelines for an organization with at least 2 examples for each.**

   Coding standards are a set of rules, guidelines, or best practices for software development. Examples include indentation and naming conventions. Guidelines are general instructions, suggestions, or guidelines for writing code. Examples include documentations and commenting.

9. **How do you enable software to be testable? Write briefly about any six of these techniques.**

   Techniques to enable software to be testable include:

   • Writing tests that are independent of one another.

   • Making sure that tests are deterministic.

   • Writing tests that test the smallest amount of code possible.

   • Utilizing mocking to isolate the system under test.

• Writing tests for each layer of the application.

• Utilizing test-driven development (TDD).

10. **Discuss any 6 coding guidelines which you are familiar with**

Six coding guidelines which I am familiar with are:

• Keep your code readable and understandable.

• Use meaningful names for variables, methods, and classes.

• Use consistent indentation and spacing.

• Avoid repeating code.

• Use comments to document and explain code.

• Write unit tests to ensure code is functioning properly.

11. **A product PQR has been released to the market characterized by the number 5.7.6. The code for this is in a directory $HOME/Project/released. There has been work continuing on this product and the .c and .h files are in a directory $HOME/Project/Module. In this scenario**

1. Identify the configurable items in the description above
2. Identify the Programmers directory
3. Identify the baselined directory
4. Identify the version number in the characterized string
5. Configurable items in the description include the product PQR, the directory $HOME/Project/released, and the directory $HOME/Project/Module.
6. The Programmers directory is $HOME/Project/Module.
7. The baselined directory is $HOME/Project/released.
8. The version number in the characterized string is 5.7.6.

12. **What is Requirement traceability Matrix? What is the need for the Same? Illustrate an RTM.**

Requirement Traceability Matrix (RTM) is a document that maps and traces user requirements with test cases. RTM is created to ensure that all the requirements are covered in the test cases. It is used to check the completeness of the test coverage and helps to keep track on the changes made to the requirements during the SDLC.

Example:

Requirement No.	Test Case ID	Result
R1	TC1	Pass
R2	TC2	Fail

13. ## Discuss any four approaches which can be followed for programming for testability

Four approaches which can be followed for programming for testability are:

• **Designing and coding for testability:** Writing code that is easy to test and debug.

• **Automating tests:** Using software tools to automate tests.

• **Creating test harnesses:** Creating test harnesses that can be used to test multiple components.

• **Using debugging tools:** Using debugging tools to find and fix bugs quickly.

14. ## Discuss any Design Pattern using its Intent, Description and an Example

Design Pattern:

Intent: To create objects that can be used in a variety of contexts, while keeping the code simple and maintainable.

Description: The Factory Method pattern is a creational design pattern that uses factory methods to create objects. It defines an interface for creating objects, but lets subclasses decide which class to instantiate.

Example:

```
public abstract class AnimalFactory {
  public abstract Animal createAnimal();
}

public class DogFactory extends AnimalFactory {
  @Override
  public Animal createAnimal() {
    return new Dog();
  }
}

public class CatFactory extends AnimalFactory {
  @Override
  public Animal createAnimal() {
    return new Cat();
  }
}
```

15. **In the following piece of C code, (ignore the line numbers 1 to 6) identify at least 5 coding practices which could make the code to be readable.**

```
int uglyduck (int t1; int array[], int I)
{
   int i;
   for (i; i < l; i++) if (num == array[i]) break;
   if (i == 1) return -1; else return i;
}
```

Modify the code with the guidelines identified to show how it will make it readable.

Coding practices that could make the code more readable:

• Use meaningful variable and function names.

• Use whitespace and indentation.

• Add comments.

• Break down complex statements into multiple lines.

• Use consistent naming conventions.

Modified Code:

```
int find_num(int target, int array[], int length)
{
   int i;
   // loop through the array to find the target number
   for (i; i < length; i++)
   {
      if (target == array[i])
      {
            break;
      }
   }
   // check if the target was found
   if (i == 1)
   {
      return -1;
   }
   else
   {
      return i;
   }
}
```

14. Discuss the Requirements Phase of a Software Development Lifecycle including a brief (2 line) description of the activities involved

The Requirements Phase of a Software Development Lifecycle involves understanding the project, collecting and analyzing the customer's needs, and developing a detailed requirements document. This document outlines the project's purpose, scope, and timeline, as well as any technical and business requirements. The requirements document serves as the foundation for the rest of the development process, and is used to create a design and build the software.

The Requirements Phase of a Software Development Lifecycle involves activities such as:

• **Defining the scope of the project** - what the software should do and how it should do it.

• **Gathering requirements from stakeholders** - understanding their needs and wants for the project.

• **Analyzing and documenting the requirements** - creating a clear document outlining what the software should do.

• **Validating the requirements** - ensuring that all requirements are accurate and meet the desired objectives.

15. Define Version, Revision and Release. A product is characterized as 4.6.2 what does this signify

Version: A version refers to a particular iteration of a software product. It is used to identify a major change in the software.

Revision: A revision is a minor change in the software and is used to identify bug fixes, small updates, and minor changes.

Release: A release refers to a particular version of the software product that is made available to the public.

The product 4.6.2 refers to a release of version 4, with a revision of 6.

16. What is CBSE? Discuss the different phases of CBSE

CBSE stands for Component-Based Software Engineering. It is an iterative software development process, which focuses on the development of components. The different phases of CBSE are:

• **Requirements:** Gather requirements, analyze, and document.

• **Design:** Create a system architecture, develop and document design.

• **Implementation:** Implement components and test for functionality.

• **Integration:** Integrate components into the system and test.

• **Deployment:** Deploy the system and monitor performance.

17. **Name two metrics which you would use in development and two metrics which you would use in the testing phases of the software development lifecycle**

    Metrics used in development:

    • Lines of code (LOC)

    • Cyclomatic complexity

    Metrics used in testing:

    • Test coverage

    • Defect density

18. **Contrast**

    1. **Coding Rules and Coding Guidelines**

       Coding rules are specific requirements that must be followed when writing code. Coding guidelines, on the other hand, are recommendations on how to write code, but may not be mandatory.

    2. **Code Inspection and Reviews**

       Code inspections are formal reviews of code by experienced developers or testers. Reviews are informal discussions about code, and are usually done in a group setting.

    3. **Packaging and Install Tools**

       Packaging tools are used to create packages of software for distribution and installation. Installation tools are used to install the packages onto a user's system.

    4. **Resolved and Closed State of Bug in a Bug Tracking System**

       Resolved means that the bug has been fixed, but has not been tested yet. Closed means that the bug has been tested and verified as fixed.

19. **Distinguish between Black-box and White-box testing? Indicate two advantages and disadvantages of both? When would you use each of them?**

    Black-box testing involves testing a system or application without any knowledge of its internal workings.

    - Advantages include the ability to focus on external system behavior and the ability to test the system from the user's perspective.
    - Disadvantages include the inability to test internal logic and the potential for missing errors.

    White-box testing involves testing a system or application with knowledge of its internal workings.

    - Advantages include the ability to test internal logic and the potential for finding more errors.

- Disadvantages include the need for detailed knowledge of the system's code and the potential for missing errors due to incomplete knowledge.

20. **Describe (1-2 sentences) 5 principles/techniques which influence a good design**

    Five principles/techniques which influence a good design are:

    • **Balance:** Achieving a sense of equilibrium among the elements in a design.

    • **Contrast:** Creating visual differences to draw attention to important elements.

    • **Repetition:** Using the same elements throughout a design to create consistency.

    • **Hierarchy:** Creating a logical order of importance to emphasize key elements.

    • **Proximity:** Grouping related elements together to create visual relationships.

MCQ - Pick the odd one out

1. **Software Design**

    a) Is a prescriptive process
    b) Outcome is the same as the process
    c) Problems have a clear true or false solution
    d) Has no stopping rules

2. **The following are examples of Architectural conflicts**

    a) Using large grain components improves performance but reduces maintainability
    b) Introducing redundant data improves availability but can reduces integrity
    c) Localizing safety degrades performance
    d) Improved depth of security increases Usability

3. **Which of the following design methodologies first specifies the individual base elements of the system in great detail?**

    a) Top down design
    b) Bottom up design
    c) Component-based design
    d) Pattern-oriented design

4. **Estimation in most scenarios is best started with estimating**

    a) Effort
    b) Schedule
    c) Cost
    d) People

5. **Discontinuance of a product is more associated with**

    a) Stopping of Sales and Support of the product
    b) Stopping the sales of the product
    c) Stopping the support of the product
    d) Stopping customers from using the product

Answers:

1. **c) Problems have a clear true or false solution** - This option does not align with the definition of software design as it is a prescriptive process that focuses on the design of a system, not the solution to specific problems.
2. **d) Improved depth of security increases Usability** - This option does not align with the definition of architectural conflicts as it does not involve trade-offs or tensions between different design decisions.
3. **b) Bottom up design** - This design methodology starts with the individual components and builds up to the overall system, while the other options focus on the overall structure first.
4. **b) Schedule** - Estimation in most scenarios should start with estimating the effort required to complete the work, as this provides a basis for determining the schedule and cost.
5. **d) Stopping customers from using the product** - Discontinuance of a product involves stopping the sales and support of the product, not just stopping customers from using it.

Unit 4

Subjective

1. Contrast the following with what is it and one example

1. Static testing and Dynamic testing

- **Static Testing:** Testing of software without executing it.
  Example: Code reviews, syntax checking.
- **Dynamic Testing:** Testing of software by executing it.
  Example: Unit testing, system testing.

2. Black-box testing and White-box testing

- **Black-box testing:** Testing of software without understanding the inner workings of the code.
  Example: Functional testing.
- **White-box testing:** Testing of software by understanding the inner workings of the code.
  Example: Code coverage testing.

3. Manual testing and Automated testing

- **Manual testing:** Testing of software by manually performing the tasks.
  Example: Exploratory testing.
- **Automated testing:** Testing of software by using automated tools.
  Example: Unit testing.

4. Process capability, Process performance and Process maturity

- **Process capability:** Ability to produce suitable results within the required parameters.
- **Process performance:** Ability to produce results consistently and reliably.
- **Process maturity:** Ability to continuously improve a process.

5. Contrast Testing and Requirement Traceability Matrix

- **Contrast Testing:** A technique used to compare the output of two different systems.
  Example: Regression testing.
- **Requirement Traceability Matrix:** A technique used to trace the requirements from the design to the code.
  Example: Linking the test cases to the requirements.

6. **The four different types of Software Maintenance.**

   • **Corrective Maintenance:** Fixing issues that arise due to errors or bugs.
   • **Adaptive Maintenance:** Modifying the existing system to fit new requirements.
   • **Perfective Maintenance:** Improving the existing system by adding new features.
   • **Preventive Maintenance:** Detecting and preventing potential issues before they occur.

2. **Discuss any 5 Software metric characteristics with one line description of the same.**

   Software Metric Characteristics:

   • **Reliability:** Measures the probability that the system will not fail.

   • **Flexibility:** Measures the ability of the system to adapt to changing requirements.

   • **Maintainability:** Measures the ease with which changes can be made to the system.

   • **Portability:** Measures the ease with which the system can be moved to different platforms.

   • **Efficiency:** Measures the amount of resources consumed by the system.

3. **Discuss any 2 reasons why Software maintenance is needed? Discuss also 2 activities which are expected and 2 actions which will support Software Maintenance**

   Reasons for Software Maintenance:

   • To fix errors in the existing system.

   • To adapt the existing system to changes in the requirements.

   Expected activities:

   • Error diagnosis and correction.

   • Modification and enhancement of existing features.

   Actions to support Software Maintenance:

   • Adequate documentation of existing system.

   • Regular review and testing of existing system.

4. **What is SEI CMM? Name two reasons why you would want to use it? Enumerate its five levels.**

   SEI CMM (Capability Maturity Model) is a model which is used to assess the maturity of software development processes. Two reasons to use the SEI CMM are:

   • To identify areas for improvement in the development process.

   • To provide a roadmap for improving the development process.

   The five levels of SEI CMM are:

   • **Level 1 – Initial:** Processes are ad hoc and reactive.

• **Level 2 – Repeatable:** Processes are repeatable, but not necessarily efficient.

• **Level 3 – Defined:** Processes are documented and standardized.

• **Level 4 – Managed:** Processes are monitored, controlled, and measured.

• **Level 5 – Optimizing:** Processes are continually improved.

5. ## Discuss briefly the steps involved in a Software Maintenance Lifecycle

The steps involved in a Software Maintenance Lifecycle are:

• **Problem Identification:** Identifying and analyzing issues in the existing system.

• **Planning:** Planning the tasks required to address the identified problems.

• **Design and Implementation:** Designing and implementing the proposed solutions.

• **Testing and Verification:** Testing and verifying the proposed solutions.

• **Deployment:** Deploying the proposed solutions to the production environment.

• **Maintenance and Support:** Providing ongoing maintenance and support for the system.

6. # Discuss briefly each of the following

    1. ## Patching What, Why and types
    2. ## Static Testing - What is it with two examples
    3. ## Regression Testing
    4. ## Re-engineering
    5. ## Reverse-engineering
    6. ## Re-structuring

• **Patching:** Patching is the process of installing updates to a computer system, with the aim of fixing software bugs, improving performance and/or adding new features. There are two types of patches – security patches and feature patches.

• **Static Testing:** Static testing is a method of testing software without actually executing the code. Examples of static testing include code review and static analysis.

• **Regression Testing:** Regression testing is a method of testing software to ensure that changes or modifications have not caused unintended side effects, and that existing functionality still works as expected.

• **Re-engineering:** Re-engineering is the process of rethinking and redesigning the way a product or service is created and delivered, with the aim of improving the overall performance of the system.

• **Reverse-engineering:** Reverse-engineering is the process of analyzing a product or system to understand how it works, in order to recreate it or to create a compatible product.

• **Re-structuring:** Re-structuring is the process of reorganizing a product or system to improve its performance, scalability, and/or maintainability.

7. ## Discuss with an example what do you understand by the following

    1. ### Smoke Testing
    2. ### Regression Testing
    3. ### Localization Testing
    4. ### Startup/Shutdown Testing

• **Smoke Testing:** Smoke testing is a type of software testing that checks the most important functions of an application quickly, to make sure the application as a whole is working.

• **Regression Testing:** Regression testing is a method of testing software to ensure that changes or modifications have not caused unintended side effects, and that existing functionality still works as expected.

• **Localization Testing:** Localization testing is a type of software testing that checks an application to make sure it is functioning properly in different locales, languages, and cultural variations.

• **Startup/Shutdown Testing:** Startup/shutdown testing is a type of software testing that checks an application to make sure it is functioning properly when it starts up and shuts down.

8. ## What is Coupling and Cohesion in Design? Why are they important? What would be ideal degree to which they need be, for a good design?

Coupling and Cohesion are two key aspects of good design. Coupling is the degree of interdependence between components, while cohesion is the degree to which components are related to each other. They are important because they can have an impact on the reliability, maintainability, and scalability of a system. The ideal degree of coupling and cohesion is low, as this will result in a system that is more robust and easier to maintain.

9. ## Discuss with two sentences each, 3 ways through which would write testable code

Three ways to write testable code are:

• Separate the business logic from the presentation layer, as this makes it easier to isolate and test the business logic.

• Refactor code regularly to improve readability, making it easier to identify errors and potential areas of improvement.

• Use unit tests to ensure that code is functioning as expected.

10. **Discuss with 1-2 sentences what does it mean and when would you use the following types of testing**
    1. **Usability testing**
    2. **Boundary testing**
    3. **White Box testing**
    4. **Destructive testing**
    5. **Smoke testing**
    6. **Localization testing**
    7. **Regression testing**

• **Usability testing:** Usability testing is a type of software testing that checks an application or website to make sure it is easy to use and meets user expectations.

• **Boundary testing:** Boundary testing is a type of software testing that checks the application to make sure that it is functioning properly at the boundaries of its input.

• **White box testing:** White box testing is a type of software testing that checks the internal workings of an application or system to ensure that it is functioning as expected.

• **Destructive testing:** Destructive testing is a type of software testing that checks an application or system to make sure it can withstand extreme conditions, such as loss of power or extreme temperatures.

• **Smoke testing:** Smoke testing is a type of software testing that checks the most important functions of an application quickly, to make sure the application as a whole is working.

• **Localization testing:** Localization testing is a type of software testing that checks an application to make sure it is functioning properly in different locales, languages, and cultural variations.

• **Regression testing:** Regression testing is a method of testing software to ensure that changes or modifications have not caused unintended side effects, and that existing functionality still works as expected.

Case Study

1. **You are shopping in a large Multi Brand Retail Store, and when you reach the payment counter, you have 3 choices. If you are a new customer, you are offered a choice to open a credit card account, when you will get a 15% discount on all your purchases today, if you are an existing customer and you hold a loyalty card, you get a 10% discount on the purchases, if you have a coupon you can get 20% off today (but it can't be used with the 'new customer' discount). Discount amounts are added, if applicable.**

   If you have to write test cases for this, how many test cases would you need for the same? Justify the same. Make assumptions as necessary needed for testing and write one sample test case including all the things which you would need, which can have the test case to be executed by an uninvolved tester.

   There are potentially 9 different test cases that would need to be written in order to test the scenario. The test cases would need to verify that the discounts are applied correctly for different combinations of customer types (new customer, existing customer, and with coupon) and discounts (15%, 10%, and 20%), and that the discounts are correctly added together.

   For example, one test case could be:

   • **Test Case:** Verify Discounts For New Customer With Coupon

   • **Precondition:** New Customer With Coupon

   • **Input:** Add items to basket, select payment option of new customer with coupon and enter coupon code

   • **Expected Results:** All items in basket should receive a 20% discount

   • **Actual Results:** All items in basket receive a 20% discount

   • **Pass/Fail:** Pass

2. **Consider your company has a software product which has software components deployed across different servers, and when a problem occurs, support engineers need to trouble shoot using the information logged by the software components. Some of the characteristics of the environment are that Log sizes are large, and these logs are generated at small intervals of time, and the interest in the logs are for particular events over specific time period in the logs. You are expected to provide an Architectural solution to support quick and effective trouble shooting.**

There are potentially different architectural approaches/solutions to solve the above problem, where each of the solutions will have different trade-off's in terms of non-functional requirements. Illustrate this by providing 3 solutions for the above scenario highlighting what non-functionalities and the trade-off's therein.

• **Solution 1:** Using a centralized logging system, where all logs from all components are stored in a single server, and are indexed and stored in a way that makes it easy to access them. This will provide quick access to the logs, but the trade-off is that the storage requirements and server load for a centralized system can be high.

• **Solution 2:** Using distributed logging systems, where each component stores its own logs in its own storage. This allows for quick access to the logs, but the trade-off is that it can be difficult to search multiple distributed systems for specific events.

• **Solution 3:** Using a log aggregation system, where all logs are stored in distributed systems, and an aggregation system is used to collect, index, and search the logs. This allows for quick and easy access to logs, but the trade-off is that the setup and maintenance of an aggregation system can be complex.

3. **Choose most appropriate types of testing applicable in the following applications from the list of testing types given below:**

   Functional, System, Load, Stress, Performance, Compatibility, Security, Reliability, Availability, Regression, Usability, Acceptance

   1. **Online shopping website (like Amazon), where they offer special sale days (like Prime Days).**
   2. **A file management system (like Google Drive), where different types of files can be uploaded and viewed from multiple devices. It also needs to provide collaborative editing feature.**
   3. **Anti-Virus software that needs to be kept up-to-date (in terms of software and data related to newer vulnerabilities) frequently.**

   4. **Load, Stress, Performance, Reliability, Availability, Regression, Usability, Acceptance** - For an online shopping website that offers special sale days, it is important to test the system's ability to handle a large number of users and transactions simultaneously, as well as its ability to maintain high performance and reliability even under heavy load. It is also important to test for compatibility with different devices and browsers, and to ensure that the system continues to function correctly after changes are made to the code (regression testing). Finally, it is important to test the usability and acceptability of the system from the user's perspective.
   5. **Functional, System, Compatibility, Security, Reliability, Availability, Regression, Usability, Acceptance** - For a file management system, it is important to test the system's ability to perform its intended functions, such as uploading, viewing, and editing files. It is also important to test the system's overall performance and reliability,

as well as its compatibility with different devices and file types. Security is an important concern for a file management system, so it is important to test the system's ability to protect user data from unauthorized access. Finally, it is important to test the usability and acceptability of the system from the user's perspective.

6. **Functional, System, Security, Reliability, Availability, Regression, Acceptance** - For an anti-virus software that needs to be kept up-to-date, it is important to test the system's ability to perform its intended functions, such as detecting and removing malware. It is also important to test the system's overall performance and reliability, as well as its ability to protect the user's device from security threats. Because the software and data related to newer vulnerabilities need to be updated frequently, it is important to test the system's ability to handle these updates without disrupting its operation (regression testing). Finally, it is important to test the acceptability of the system from the user's perspective.

True or False

1. Fault based testing is a Code based testing where a fault is purposely introduced into the code
2. Acceptance Testing is performed anytime during Agile development on request of the customer
3. It is easier to execute Regression testing manually
4. Destructive model is a testing model chosen as part of testing strategy
5. Test adequacy criteria helps to determine when to stop testing
6. Bugs/kLoC is a valid test metric

Answers

1. **True** - Fault-based testing is a code-based testing technique in which a fault, or error, is purposely introduced into the code in order to test the system's ability to detect and recover from the fault.
2. **False** - Acceptance testing typically takes place at the end of the development process, after all other types of testing have been completed. In an agile development process, acceptance testing may be performed on an iterative basis, but it is not performed on request of the customer.
3. **False** - Regression testing can be performed manually or using automated tools, but it is generally easier to execute regression testing using automated tools because it involves running a large number of tests that may need to be repeated multiple times.
4. **False** - The destructive testing model is not a testing model that is typically chosen as part of a testing strategy. Destructive testing involves deliberately causing damage to a product in order to test its strength and durability, and is not typically used to evaluate the functionality of software.
5. **True** - Test adequacy criteria are used to determine when to stop testing, and are based on factors such as the amount of time and resources available for testing, the level of coverage achieved, and the risk and severity of the defects that have been found.

6. **False** - "Bugs/kLoC" is not a valid test metric. "kLoC" stands for "thousands of lines of code," and bugs per kLoC is not a useful metric for measuring the effectiveness of testing because it does not take into account the complexity or importance of the code being tested.

MCQs - Choose the one that DOES NOT belong in the group

1. The following illustrates the Maintainability Quality Attribute of a program

```
 a. Mean time to change
 b. Change requests to new version
 c. Cost to Correct
 d. Failures/hour of operation
```

2. These are good examples of Quality attributes from a Product operation perspective

```
 a. efficiency
 b. Reliability
 c. Testability
 d. Usability
```

3. Capability Maturity Model has the following levels of maturity

```
 a. Initial
 b. Planned
 c. Managed
 d. Optimized
```

4. The following fall into the boundaries of Professional Ethics

```
 a. Skill
 b. Integrity
 c. Accountability
 d. Non-Plagiarism
```

5. The following characterizes a program

```
 a) A program is for the programmer who reads it, not for the computer that runs
it
 b) Programs have "personalities": messy, verbose, cryptic, neat
 c) A program is written once, but read and modified a lot many times.
 d) Program is well written when it makes it difficult for others to modify it
```

6. "Testability" is the ease of verifying that a software product satisfies its specifications and requirements. Which of the following is NOT a tactical technique meant to enhance testability?

```
 a) Building "test points" into the code (which can be used for setting or
retrieving current module status and variable contents.)
 b) Including "test stubs" (that returns known fixed values that emulate
functions not currently under test)
 c) "Instrumenting" the code (such as adding execution logging and "I got here"
 messages so that the module under test reveals more about its operation).
 d) Add one more tester to the team.
```

7. The main difference between Verification and Validation can be explained by the following statement(s) in the order given below:

```
 a) Essentially, they are the same and can be used interchangeably
 b) Building the thing right, building the right thing
 c) Building the right thing, building the thing right
 d) Verification takes place during post the testing phases; validation takes
place during the testing phase
```

8. Characterization of an effective test case is

```
 a) When it finds issues
 b) When it can be easily executed
 c) When it has higher percentage of code coverage
 d) When it validates the features under test
```

9. A developer has been assigned to work on a defect report. Once the developer identifies and makes necessary change(s) to the relevant module(s), she has to test the software to make sure that the defect has been fixed correctly. Further, she also runs tests that have been run previously and have succeeded. This test to ensure that she has not introduced any new defects while making a code change is known as:

```
 a) Regression testing
 b) Integration testing
 c) Unit testing
 d) Acceptance testing
```

Answers:

1. **b. Change requests to new version** - This option does not belong in the group because the other options are all examples of metrics that can be used to measure the maintainability of a program, whereas "change requests to new version" is not a metric that directly measures maintainability.
2. **d. Usability** - This option does not belong in the group because the other options are all examples of quality attributes that can be measured from a product operation perspective, whereas usability is a quality attribute that is typically measured from a user perspective.
3. **d. Optimized** - This option does not belong in the group because the other options are all levels of the Capability Maturity Model, which is a framework used to evaluate the maturity of an organization's processes, whereas "optimized" is not one of the defined levels of the CMM.

4. **a. Skill** - This option does not belong in the group because the other options are all examples of ethical principles that a professional should uphold, whereas "skill" is not an ethical principle.
5. **d. Program is well written when it makes it difficult for others to modify it** - This option does not belong in the group because the other options are all true statements about programs, whereas this option is a false statement. A well-written program should be easy to read and understand, and should not be designed to make it difficult for others to modify it.
6. **d. Add one more tester to the team** - This option does not belong in the group because the other options are all tactical techniques that can be used to enhance testability, whereas adding another tester is not a technique that directly affects testability.
7. **a. Essentially, they are the same and can be used interchangeably** - This option does not belong in the group because the other options are all correct statements that describe the main differences between verification and validation, whereas this option is a false statement. Verification and validation are different processes with distinct goals and purposes, and they should not be used interchangeably.
8. **d. When it validates the features under test** - This option does not belong in the group because the other options are all characteristics of an effective test case, whereas this option is not a characteristic of an effective test case. Validating the features under test is a goal of testing, but it is not necessarily a characteristic of an effective test case.
9. **d. Acceptance testing** - This option does not belong in the group because the other options are all types of testing that can be performed on a software product, whereas acceptance testing is not a type of testing. Acceptance testing is a process in which the software is evaluated by the customer or end user to determine whether it meets their requirements and is suitable for use.

Unit 5

Subjective

1. **Name any 5 reasons on the relevance of global software development**

   Five reasons on the relevance of global software development:

   • Cost savings

   • Time-to-market advantage

   • Access to global talent pool

   • Global customer-centric approach

   • Leveraging technologies and resources from different parts of the world.

2. **Briefly describe 5 widely accepted ethical characteristics expected of a software engineer**

   Five widely accepted ethical characteristics expected of a software engineer:

   • Honesty: Presenting accurate information and data.

   • Integrity: Abiding by a code of professional ethics.

   • Objectivity: Not allowing personal preferences to affect professional decisions.

   • Respect: Respecting the rights, beliefs, and opinions of others.

   • Responsibility: Taking responsibility for one's actions.

3. **Discuss the four common themes that any team looking to implement DevOps needs to focus its time and resources on. How does this complement the DevOps pipeline?**

   The four common themes that any team looking to implement DevOps needs to focus its time and resources on are:

   1. Culture and collaboration: DevOps emphasizes collaboration and cross-functional teamwork, and a key part of implementing DevOps is building a culture that supports these principles. This includes fostering open communication, collaboration, and shared ownership across teams, as well as promoting a continuous learning and improvement mindset.
   2. Automation: Automation is a key component of DevOps, and teams need to focus on implementing tools and processes that enable them to automate repetitive and time-consuming tasks, such as building, testing, and deploying software. This not only speeds up the software delivery process, but also helps to reduce errors and improve the overall quality of the software.
   3. Measurement and feedback: In order to continuously improve, teams need to collect and analyze data on their processes and performance, and use this information to make informed decisions and adjust their practices. This requires implementing

systems and processes for measuring and collecting data on key performance indicators, as well as using this data to provide feedback to teams and individuals.

4. Continuous delivery and deployment: DevOps emphasizes the importance of delivering software frequently and reliably, and teams need to focus on implementing processes and tools that enable them to do this. This includes implementing continuous integration and continuous delivery practices, as well as automating the deployment of software to production environments.

These four themes are closely related and complement the DevOps pipeline, which is a set of practices and tools for automating and optimizing the software delivery process. By focusing on these themes, teams can build a culture and infrastructure that supports the DevOps pipeline and enables them to deliver high-quality software quickly and reliably.

4. **Discuss what are the following with one line descriptions**
    1. **Continuous Integration**
    2. **Continuous Build**
    3. **Continuous Delivery**
    4. **Continuous Testing**
    5. **Continuous Deployment**

• Continuous Integration: Automatically integrating changes made by developers in a shared code repository.

• Continuous Build: Automatically building projects with each code change.

• Continuous Delivery: Making software available to stakeholders in a timely manner.

• Continuous Testing: Automatically testing software as it is being developed.

• Continuous Deployment: Automatically deploying code changes to production.

5. **Answer the following:**
    1. **Some of the most exemplary DevOps initiatives were started in giant and mature IT organizations. True or False.**
    2. **An important ethic for a software engineer is to keep in mind the overall good the software will be for the larger society. True or False.**
    3. **Trying out solutions until an optimal solution is created is _ _ _ _ _ _ _ _ _ _ _ and working with one solution that suits all constraints and requirements is _ _ _ _ _ _ _ _ _ _ _.**

1. True
2. True
3. Trial & error / Optimization

6. **What are the challenges of Geographical distance with respect to Software Development? How can they be overcome?**

   Challenges of Geographical distance with respect to Software Development:

   • Communication issues: Difficulties in sharing information, ideas, and opinions.

   • Cultural differences: Different beliefs, values, and norms.

   • Time zone differences: Difficulty in scheduling meetings and synchronizing work.

   • Language barriers: Difficulty in understanding and explaining technical requirements.

   These challenges can be overcome by using tools such as remote communication platforms, project management tools, and automated testing tools.

7. **Describe the term DevOps. Write briefly about the DevOps pipe line.**

   DevOps is a set of practices that enables an organization to deliver software products faster and more reliably. The DevOps pipeline is a process that involves a series of steps that are taken to ensure that software is released quickly and efficiently. The pipeline includes steps such as development, testing, deployment, and monitoring.

8. **Define Metric and Measures with respect to Software Quality. Name one example measure for the following quality attributes of a product. Correctness, Maintainability, Integrity & Usability**

   Metrics and Measures are used to measure the quality of a software product.

   • Correctness: Number of defects in the product per line of code.

   • Maintainability: Number of bugs reported in the product over time.

   • Integrity: Number of security vulnerabilities found in the product.

   • Usability: Number of user complaints about the product.

9. **Discuss briefly**

   1. **Process Capability, Process Performance and Process maturity**
   2. **Professional Ethics with 2 examples of professional ethical practices**
   3. **Any 3 reasons which makes global software development to be relevant**

   1. Process Capability: It is the ability of a process to produce products that meet customer requirements and expectations.

      Process Performance: It is the degree to which a process meets its goals.

Process Maturity: It is the degree to which a process is able to achieve its desired outcomes.

2. Professional Ethics: Professional ethics refers to the ethical principles that guide the behavior of individuals and organizations in their professional lives. Examples of professional ethical practices include:
   - Honesty: Being truthful in all professional dealings.
   - Integrity: Abiding by a code of professional ethics.
   - Respect: Respecting the rights, beliefs, and opinions of others.

3. Reasons which makes global software development to be relevant:
   - Cost effectiveness
   - Access to global talent
   - Leveraging technologies and resources from different parts of the world
   - Time-to-market advantage
   - Global customer-centric approach

10. **Discuss the four common themes that any team looking to implement DevOps needs to focus its time and resources on. How does this complement the DevOps pipeline?**

    The four common themes that any team looking to implement DevOps needs to focus its time and resources on are: Automation, Collaboration, Measurement & Monitoring, and Security.

    These themes complement the DevOps pipeline by providing the necessary tools and processes to ensure that the software is released quickly, securely, and reliably.

11. **Discuss what is Patching? What do you understand by Hot and Cold patching? Where does this fit in the Maintenance Lifecycle?**

    Patching is the process of updating software with bug fixes, security updates, or new features.

    Hot patching is the process of applying a patch to a running system without restarting it. Cold patching is the process of applying a patch to a system that is not running.

    Patching fits into the Maintenance Lifecycle in the stage of corrective maintenance, which is the process of fixing problems and bugs in the software.

12. **Discuss any 4 reasons why an Organization would like to look at Global Development? Discuss any 4 challenges associated with Global Development.**

Reasons why an organization would like to look at Global Development:

• Cost savings

• Access to global talent pool

• Leveraging technologies and resources from different parts of the world

• Time-to-market advantage

• Global customer-centric approach

Challenges associated with Global Development:

• Cultural differences and language barriers

• Different legal regulations in different countries

• Different time zones

• Difficulty in coordinating and managing distributed teams

• Difficulty in establishing trust between teams

13. **List at least 2 metrics which you would look for understanding the quality of each of the SDLC phases of requirements, design, implementation, testing and overall project.**

Requirements: Number of requirements vs. number of requirements implemented.

Design: Number of design defects vs. number of design defects resolved.

Implementation: Number of code defects vs. number of code defects resolved.

Testing: Number of test cases failed vs. number of test cases passed.

Overall Project: Number of defects vs. number of defects resolved.

14. **Discuss the set of activities involved in software maintenance**

The activities involved in software maintenance include:

• Corrective Maintenance: Fixing existing errors and bugs

• Adaptive Maintenance: Modifying the software to adapt to changes in the environment

• Perfective Maintenance: Enhancing the software with new features and capabilities

• Preventive Maintenance: Performing preventive measures to avoid errors and reduce downtime

15. **Name 5 reasons which support the need of Global Software Development**

    Five reasons which support the need of Global Software Development:

    • Cost savings

    • Time-to-market advantage

    • Access to global talent pool

    • Leveraging technologies and resources from different parts of the world

    • Global customer-centric approach.

16. **Explain in 4-5 sentences**

    1. **Measure and Metrics**
    2. **Hacking**
    3. **SOA**

    Measure and Metrics: Quantitative measures that provide data about the characteristics of a product.

    Hacking: Using non-traditional means to quickly solve problems.

    SOA: Service-oriented architecture, a style of software design in which services are provided to the other components by application components through a communication protocol over a network.

17. **Contrast the following**

    1. **COCOMO and Delphi Estimation techniques**
    2. **V Model and Waterfall Model**
    3. **Functional and Non-functional requirements**
    4. **Reviews and Inspection**

    COCOMO and Delphi Estimation techniques: COCOMO is a model based on cost drivers and uses a bottom-up approach. Delphi is a model based on expert opinion and uses a top-down approach.

    V Model and Waterfall Model: V Model is an iterative development model that combines the waterfall model and iterative model. Waterfall model is a sequential development model and is used in small projects.

    Functional and Non-functional requirements: Functional requirements specify what the system should do, while non-functional requirements specify how the system should perform.

    Reviews and Inspection: Reviews are informal meetings to discuss the product, while inspections are formal meetings to detect errors in the product.

18. ## List out

    1. ### Four (4) reasons why global software development is relevant
    2. ### Five (5) levels of Software Capability Maturity (SEI CMM)
    3. ### Four (4) Ethical characteristics expected of a software Engineer

    Four (4) reasons why global software development is relevant: Cost savings, Time-to-market advantage, Access to global talent pool, Leveraging technologies and resources from different parts of the world.

    Five (5) levels of Software Capability Maturity (SEI CMM): Initial, Repeatable, Defined, Managed, and Optimizing.

    Four (4) Ethical characteristics expected of a software Engineer: Honesty, Integrity, Objectivity, Respect, and Responsibility.

MCQs - Choose the MOST appropriate one which DOES NOT belong to the group

1. Software maintenance happens due to

```
 a. Correcting of faults in the software b. Improving of the design of the
software
 c. Implementation of new features
 d. need for generating revenues from the software
```

2. Hacking is closer to

```
 a. Using non-traditional means to quickly solve problems
 b. Systematic problem solving c. Abusing the system for solving a problem
 d. Illegal way of solving a problem.
```

3. Ethical characteristics for a good software developer are characterized with law abiding

```
 a. Adherence to characteristics like Integrity, Honesty, Accountability,
 b. Not plagiarizing work or Intellectual property
 c. Ensuring that you deliver at whatever cost/approach to what is being measured
d. Ensuring confidentiality of resources and work as expected from the
organization
```

4. Availability of a product is characterized by

```
 a. Can I use it
 b. Can I use it wherever! want it
 c. Can I use it whenever I want it
 d. Will I have the permissions to use it 5. Software Reliability has
relationship with
 a. Mean Time Between Failure (MTBF)
 b. Failure Rate
 c. Fault Density
 d. Mean Time for recovery
```

Answers

1. d. need for generating revenues from the software
2. c. Abusing the system for solving a problem
3. d. Ensuring confidentiality of resources and work as expected from the organization
4. d. Will I have the permissions to use it
5. a. Mean Time Between Failure (MTBF)

MCQ - Choose the most appropriate one in the group

1. The following characterizes a Hacker

```
 a. Cleverness to move ahead in spite of obstacles or challenges
 b. Capability to explore and exploit weakness of system
 c. Unethical and with criminal orientation
 d. Approaches solutions in un-conventional means
```

2. The following are the Deployment goals for Applications and Services a. Consistent and Successful Deployment

```
 b. Support increase in frequency of Deployment
 c. Support faster time to Market
 d. All of the Above
```

3. DevOps

```
 a. Has no influence on the quality of the product
 b. Extends Delivery to the Operations team and Operations feedback to the
Delivery team
 c. Achieves faster deployment of products to market
 d. Promotes better collaborations between Dev and Operations
```

4. ITIL

```
 a. Describes a set of best practices for IT Service Management processes
 b. Is organization and technology specific
 c. Is available as an IT Service Lifecycle
 d. Supports effective utilization of the IT Infrastructure for Stability,
Scaling, Responsiveness
```

Answers

1. c. Unethical and with criminal orientation
2. d. All of the Above
3. a. Has no influence on the quality of the product
4. d. Supports effective utilization of the IT Infrastructure for Stability, Scaling, Responsiveness