**1.What is Software configuration Management ? Difference between stub and drivers? (3+2)**

Software Configuration Management (SCM) is a set of processes and tools used to systematically manage, control, and track changes in software development projects. SCM encompasses version control, change management, and release management to ensure that software is developed, tested, and delivered in a consistent and organized manner.

The difference between stubs and drivers in the context of software testing is that:

1. Stubs are used in bottom-up integration testing. They simulate lower-level modules that have not yet been developed or are not available for testing. Stubs provide a way to test higher-level modules without their full implementations.
2. Drivers, on the other hand, are used in top-down integration testing. They simulate higher-level modules to test lower-level modules that depend on them. Drivers help in testing the interactions of lower-level modules with higher-level modules that are not yet fully implemented.

In summary, stubs are used to simulate missing lower-level components in bottom-up testing, while drivers simulate higher-level components in top-down testing. Both are essential for integration testing to ensure that different parts of a software system work together as intended.

**2.Black box testing vs white box testing? Difference between Material requirement plan(MRP) and enterprise resource planning (ERP) ? (3+2)**

1. Black Box Testing vs. White Box Testing:

- Black Box Testing is a software testing method where the internal structure, logic, and code of the software being tested are not known to the tester. Testers focus on input and output behavior to verify the correctness of the software.

- White Box Testing, on the other hand, is a testing method where the tester has full knowledge of the internal code, structure, and logic of the software. Testing is done with an understanding of the code to ensure it's working correctly from a structural perspective.

II. Material Requirement Planning (MRP) vs. Enterprise Resource Planning (ERP):

- MRP is a system that focuses on managing and planning the materials needed for production. It helps in determining what materials are required, in what quantities, and when they are needed to meet production goals.

- ERP is a broader system that integrates various aspects of a business, including MRP. ERP encompasses not only material planning but also other functions like finance, human resources, sales, and more. It provides a holistic view of business processes and helps in managing resources across the organization.

The key difference is that MRP is primarily concerned with material planning, while ERP is a comprehensive system that covers various aspects of business operations beyond just materials.

**3.Explain Customer relationship management? Explain UML? (2+3)**

1. **Customer Relationship Management (CRM):**

Customer Relationship Management (CRM) is a strategic approach and a set of practices, often supported by technology, that organizations use to manage and analyze their interactions with customers and potential customers. The primary goal of CRM is to build and maintain strong and lasting relationships with customers, leading to improved customer satisfaction, loyalty, and business success. It involves collecting and managing customer data, analyzing customer behavior and preferences, and using this information to enhance customer interactions, marketing efforts, and overall customer experience.

II. **Unified Modeling Language (UML):**

Unified Modeling Language (UML) is a standardized modeling language used in software engineering to visually represent and document the design and structure of software systems. UML provides a set of graphical notations for creating diagrams that depict different aspects of a software system, such as its structure, behavior, and interactions. It is widely used to communicate and document software designs, making it easier for software development teams to understand, collaborate on, and implement complex software systems. UML diagrams include various types like class diagrams, use case diagrams, sequence diagrams, and more, each serving a specific purpose in software design and development.

1. **Different between ISO and CMM1? Top down and bottom up approach? (3+2)**
2. ISO (International Organization for Standardization) and CMMI (Capability Maturity Model Integration):

- ISO (International Organization for Standardization): ISO is an international body that develops and publishes standards for various industries. ISO standards, such as ISO 9001 (Quality Management) and ISO 27001 (Information Security), provide guidelines and best practices for organizations to ensure quality, safety, and efficiency in their processes and products.

- CMMI (Capability Maturity Model Integration): CMMI is a process improvement approach that focuses on the maturity of an organization's processes. CMMI provides a framework for assessing and improving an organization's capability to develop and maintain quality products and services. It is commonly used in the software and systems engineering industry.

II. Top-Down vs. Bottom-Up Approach:

- Top-Down Approach: In a top-down approach, you start with a broad, high-level perspective and gradually break it down into more detailed components. This approach is often used in software design and project planning. For example, when designing a software system, you would first create an overall architecture or design, and then progressively decompose it into smaller modules or components.

- Bottom-Up Approach: A bottom-up approach begins with the detailed components and assembles them to form a larger, more comprehensive system. This approach is commonly employed in software development during the integration phase. For instance, in integration testing, you start with individual modules or components and combine them to test their interactions and functionality in the context of the whole system.

These approaches are used in various aspects of software development and project management, depending on the specific needs of a project and the desired level of detail and control.

1. **Explain types of software testing. 5**
2. **Unit Testing:** Unit testing involves testing individual components or units of code in isolation. The goal is to ensure that each unit functions correctly. Developers typically perform unit testing to validate the correctness of their code.
3. **Integration Testing:** Integration testing checks how different components or modules of a software system work together when integrated. It aims to identify and resolve issues related to the interaction between these components.
4. **Functional Testing:** Functional testing verifies that a software application's functions and features work according to specified requirements. Testers assess the software's behavior based on input and expected output.
5. **Regression Testing:** Regression testing is conducted to ensure that new code changes or updates have not introduced new defects or caused existing features to break. It involves retesting previously validated functionality.
6. **Performance Testing:** Performance testing assesses a software application's speed, scalability, and responsiveness under different conditions, such as varying loads and user interactions. Types of performance testing include load testing, stress testing, and scalability testing.
7. **Security Testing:** Security testing checks the software's vulnerability to security threats and ensures that it can resist unauthorized access, data breaches, and other security risks.
8. **Usability Testing:** Usability testing evaluates the software's user-friendliness, including its user interface, navigation, and overall user experience. The goal is to ensure that the software is intuitive and easy to use.
9. **Compatibility Testing:** Compatibility testing ensures that the software functions correctly on various platforms, devices, and web browsers. It helps identify issues related to cross-browser compatibility, OS compatibility, and hardware requirements.
10. **User Acceptance Testing (UAT):** UAT is typically the final phase of testing before a software release. End-users or stakeholders validate whether the software meets their requirements and is ready for production use.
11. **Alpha and Beta Testing:** Alpha testing is performed by a select group of internal testers within the development organization. Beta testing involves external users who provide feedback on the software before its public release. These tests help uncover real-world issues.

These are just some of the many types of software testing. The choice of testing type depends on the project's objectives, the software's characteristics, and the specific aspects that need to be validated.