Differentiate between white Box retesting and Black Box testing with both advantage and disadvantage.

White-Box Retesting

Definition: Testing with knowledge of the internal code.

Advantages:

1. **Detailed Testing:** Finds hidden bugs.
2. **Improves Code:** Helps optimize performance.
3. **Complete Coverage:** Tests all code paths.

Disadvantages:

1. **Requires Code Knowledge:** Tester must understand the code.
2. **Time-Intensive:** More time needed to test all paths.
3. **Maintenance:** Tests may need updates with code changes.

Black-Box Testing

Definition: Testing without knowing the internal code.

Advantages:

1. **User-Focused:** Tests how the software works for users.
2. **Easy to Use:** No need to know code details.
3. **Broad Testing:** Covers overall functionality.

Disadvantages:

1. **Partial Coverage:** May miss some internal details.
2. **Harder to Fix Bugs:** Harder to locate issues.
3. **Repetition:** Might test the same features multiple times.

Using both methods provides a fuller picture of software quality.

Show the logical organization of esting List out the test plan attributes with test execution and reporting scenario.

Logical Organization of Testing

1. **Plan:** Decide what to test and how.
2. **Design:** Create test cases and prepare data.
3. **Execute:** Run the tests and check results.
4. **Report:** Share test results and issues.

Test Plan Attributes

1. **Objectives:** What you want to achieve.
2. **Scope:** What will be tested.
3. **Resources:** Who and what is needed.
4. **Schedule:** When testing will happen.
5. **Criteria:** How you'll know if tests pass or fail.
6. **Risks:** Possible problems and solutions.
7. **Deliverables:** Reports and logs you'll provide.

Test Execution and Reporting Scenario

1. **Execution:** Perform tests and record results.
2. **Reporting:** Document and share what you found and any issues.

How to measure software reliability? Give a proper example.

Measuring Software Reliability

1. **Defect Density:**
   - **Formula:** $\text{Defect Density} = \frac{\text{Number of Defects}}{\text{Size of Software}}$
   - **Example:** 50 defects in 10,000 lines of code = 0.005 defects per line.
2. **Mean Time to Failure (MTTF):**
   - **Formula:** $\text{MTTF} = \frac{\text{Total Operating Time}}{\text{Number of Failures}}$
   - **Example:** 500 hours of operation with 5 failures = 100 hours between failures.
3. **Mean Time to Repair (MTTR):**
   - **Formula:** $\text{MTTR} = \frac{\text{Total Repair Time}}{\text{Number of Repairs}}$
   - **Example:** 10 hours of repair time for 5 repairs = 2 hours to fix a failure.
4. **Availability:**
   - **Formula:** $\text{Availability} = \frac{\text{Total Uptime}}{\text{Total Uptime} + \text{Total Downtime}}$
   - **Example:** 950 hours uptime and 50 hours downtime = 95% availability.

These metrics help assess how reliable software is by measuring defects, failure intervals, repair times, and operational uptime.

What is Six-Sigma methodology for software engineering? Define the role of SQA group.

Six-Sigma Methodology in Software Engineering
Definition: A method to improve quality by reducing defects and variations, aiming for fewer than 3.4 defects per million opportunities.
Key Steps:

1. **Define:** Identify the problem.
2. **Measure:** Collect data.
3. **Analyze:** Find root causes.
4. **Improve:** Fix issues.
5. **Control:** Maintain improvements.

Role of the SQA Group
Definition: Ensures software meets quality standards.
Key Responsibilities:

1. **Plan:** Set quality goals.
2. **Monitor:** Check adherence to processes.
3. **Test:** Find and fix defects.
4. **Improve:** Suggest process enhancements.
5. **Ensure Compliance:** Follow standards and regulations.

Write down the principie of agile method with its drawback.

Principles of Agile in Software

1. **Frequent Delivery:** Release working software regularly.
2. **Embrace Changes:** Adjust to new requirements easily.
3. **Regular Updates:** Deliver updates at short intervals.
4. **Collaborative Teams:** Work closely with team members.
5. **Motivated Individuals:** Build around motivated people.
6. **Face-to-Face Talks:** Prefer direct communication.
7. **Working Software:** Measure progress by functional software.
8. **Consistent Work:** Maintain a steady work pace.
9. **High Quality:** Focus on technical excellence and design.
10. **Simplicity:** Do only what's necessary.
11. **Self-Organizing Teams:** Let teams manage their tasks.
12. **Reflect and Improve:** Regularly review and enhance processes.

Drawbacks of Agile in Software

1. **Less Documentation:** May lead to incomplete documentation.
2. **Scope Creep:** Frequent changes can extend project scope.
3. **Resource Intensive:** Requires significant ongoing effort.
4. **Communication Issues:** Can be challenging for remote teams.
5. **Unpredictable Results:** Outcomes may vary without detailed upfront planning.

Show the waterfall model with i: phases and problem. Define evolutionary development.

Waterfall Model

Phases:

1. **Requirements:** Gather all requirements.
2. **Design:** Plan the system design.
3. **Implementation:** Write the code.
4. **Testing:** Test the system.
5. **Deployment:** Release the software.
6. **Maintenance:** Fix issues and update.

Problems:

- **Rigid:** Hard to change once a phase is done.
- **Late Testing:** Issues are found too late.
- **Fixed Requirements:** Assumes requirements won't change.

Evolutionary Development

Definition: Build software in small, iterative chunks. Each version is improved based on feedback and changing needs.

Key Points:

- **Iterative:** Develop in repeated cycles.
- **Incremental:** Add features gradually.
- **Feedback-Based:** Adjust based on user input.

At the end of their study program, students in a software engineering course are typically expected to complete a major project. Explain how the agile methodology may be very useful for the students to use in this case.

Agile for Student Projects
Main Theme: Agile helps students manage projects by breaking them into small parts and improving them based on feedback.
Benefits:

1. **Frequent Feedback:** Regular input helps make early adjustments.
2. **Iterative Work:** Allows working in small, manageable cycles.
3. **Clear Goals:** Sets specific objectives for each cycle.
4. **Team Collaboration:** Encourages teamwork and communication.
5. **Flexibility:** Adapts to changes and new ideas easily.

What is component-based software engineering? Define spiral model of software process with its application in different sectors
Component-Based Software Engineering (CBSE)
Definition: Building software using reusable, pre-tested components.
Main Theme: Focuses on assembling software from modular, reusable pieces to speed up development and improve quality.
Spiral Model of Software Process
Definition: An iterative process combining planning, risk analysis, development, and evaluation in repeated cycles.
Applications:

1. **Software Development:** Complex projects with changing requirements.
2. **Defense:** Risk management in defense systems.
3. **Aerospace:** Managing complex, high-reliability projects.
4. **Healthcare:** Adapting to evolving medical software needs.
5. **Finance:** Handling risks and changes in financial systems.

Main Theme: The Spiral Model is ideal for managing risks and adapting to changes through iterative development.