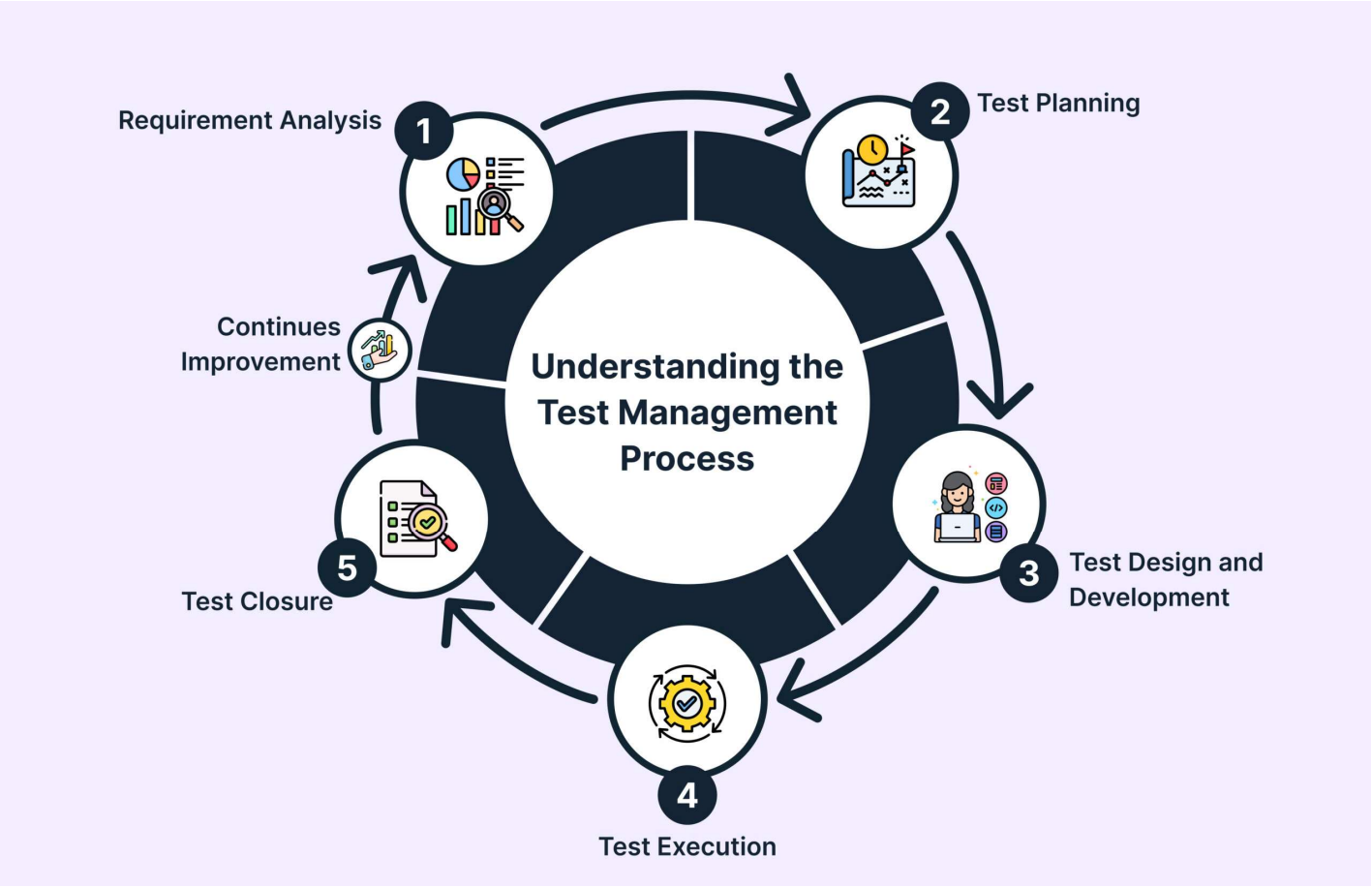
Key elements of test management

What is Test Management?

Test Management is a process that requires proactive managing activities related to the testing of software. It involves planning, organizing, coordinating, and controlling testing activities for a software development project.



1. Requirement Analysis

Understanding the requirements and establishing clear testing objectives before starting the process.

- **Risk Analysis:** Identifying and assessing potential risks in the project at various levels.
- **Test Estimation:** Determining the effort and resources required for testing.

2. Test Planning

Define the overall testing strategy and approach by understanding project requirements and objectives.

- **Define Scope and Pre-requisites:** Identify test levels like Unit Testing, Integration Testing, System Testing, etc. Determine testing resources like tools, and environments.
- **Document a test plan** outlining the testing strategy, scope, schedule, and deliverables.

3. Test Design & Development

Create test cases with detailed test steps based on the Test plan. Write Test Scripts for the test cases.

- **Define Test Scenarios:** Identify test scenarios based on the business requirements and specifications.
- **Create Test Cases:** Based on the test scenarios, create test cases that can check the application on different levels i.e. functional and non-functional.
- **Write Test Steps**: For the given test cases, write detailed and clear test steps that should be followed to run the test cases. Mention the expected results for each test case and compare actual results to declare a test case pass or fail.
- **Write Test Scripts:** For automated tests, write test scripts based on the given test cases to run in the selected test framework such as Selenium, Cypress, Appium.

4. Test Execution

Set up testing environment and execute test cases using prescribed frameworks, tools, and techniques.

- **Set up Environment:** Configure test tools and set up hardware, software, and network configurations.
- **Execute Test Cases:** Run test cases manually or using automated testing tools, record test results and document any defects found.
- **Monitor and manage test execution:** Ensure faster test execution with Test Case Prioritization, where crucial tests are run before others.

5. Test Closure

Track defects, consolidate testing activities in Test Execution Report, and find opportunities on continuous improvement.

- **Track Defects:** Log all the defects, prioritize them as per criticality and track them to resolve defects efficiently.
- **Create Test Report:** Summarize testing activities in proper reports and evaluate test completion against the test plan.

Continuous Improvement (Across All Phases)

This final element indicates that testing is **iterative**.

- Each cycle improves tools, processes, standards, documentation, test cases, and execution methods.
- Uses techniques like **Root Cause Analysis (RCA)**, metrics analysis, and feedback.
- Helps make future testing faster, more efficient, and more accurate.

Explain test organization and structure of testing group.

⭐ **Short Note on Test Organization**

Test Organization refers to the way a software testing team is structured, managed, and coordinated within a project or company. It defines **roles, responsibilities, reporting levels, and workflow**, ensuring that testing activities are performed efficiently and systematically.

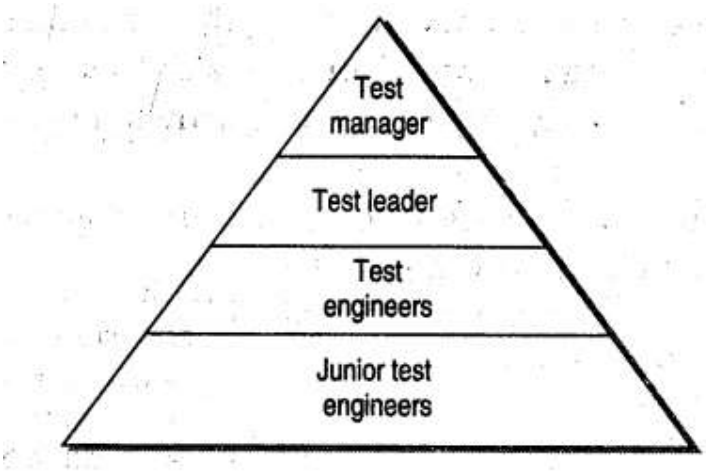
A well-organized test structure enables proper planning, control, communication, and independence of the testing function.

Key Points:

1. **Defines Testing Roles**
    - Includes Test Manager, Test Lead, Test Engineers, Junior Testers, Automation/Test Architects.
    - Clearly specifies responsibilities and authority.
2. **Establishes Reporting Structure**
    - Typically a pyramid hierarchy:
      Test Manager → Test Leader → Test Engineers → Junior Test Engineers.
    - Ensures smooth communication and accountability.
3. **Supports Independence in Testing**
    - Testers are separated from developers to avoid bias.
    - Independent testing leads to higher software quality.
4. **Allocates Resources and Tools**
    - Test organization manages manpower, test environments, tools, and infrastructure required for testing.
5. **Defines Processes and Standards**
    - Establishes uniform test processes, defect reporting procedures, templates, and documentation standards.
6. **Ensures Coordination Between Teams**
    - Acts as a bridge between development, project management, business analysts, and stakeholders.
7. **Improves Efficiency and Quality**
    - A structured test organization ensures systematic work, reduces defects, and improves productivity.

Short Summary (2–3 lines)

Test Organization defines how the testing team is structured and managed. It establishes roles, processes, and communication channels to ensure effective, independent, and high-quality testing.



Testing Group Hierarchy

Testing Group Hierarchy (Explained with Diagram)

The diagram represents a **pyramid structure** of the testing organization.
It shows how responsibilities and authority are distributed from top to bottom.

---

1) Test Manager (Top Level)

The highest authority in the testing group.

Roles & Responsibilities:

1. Defines overall **testing strategy and goals**.
2. Prepares **master test plan** and estimates budget, time, and resources.
3. Coordinates with project manager, development manager, business team.
4. Manages high-level risks and resolves escalations.
5. Approves test processes, standards, tools, and training.
6. Reviews test metrics, reports, and ensures quality of testing deliverables.
7. Manages test environment, resource allocation, and release decisions.
8. Provides final approval for **Test Summary Report**, test closure.
9. Leads continuous improvement of testing processes.

---

2) Test Leader / Test Lead (Middle Level)

Reports to the Test Manager and acts as the bridge between manager and testers.

Roles & Responsibilities:

1. Converts the test plan into **task assignments**.
2. Conducts daily stand-up meetings with the test engineers.

3. Prepares detailed schedules for test execution.
4. Allocates work to test engineers based on skill and experience.
5. Reviews test cases, scenarios, scripts prepared by testers.
6. Monitors test progress and defect trends.
7. Coordinates with developers to resolve defects and clarifications.
8. Ensures that test environments and test data are ready.
9. Prepares daily/weekly status reports for the test manager.

---

3) Test Engineers (Core Execution Level)

These are the **main testers** responsible for performing testing activities.

Roles & Responsibilities:

1. Write detailed **test cases**, **test scenarios**, and **test data**.
2. Execute test cases manually or through automation tools.
3. Log defects in bug tracking tools (JIRA, Bugzilla, Azure DevOps).
4. Perform **retesting and regression testing**.
5. Verify requirement coverage through RTM.
6. Identify edge cases and perform exploratory testing.
7. Maintain test logs and update daily progress.
8. Communicate with developers regarding defects and clarifications.
9. Ensure adherence to testing standards and procedures.

---

4) Junior Test Engineers (Entry Level)

They are new team members with limited experience and usually work under the guidance of test engineers or test lead.

Roles & Responsibilities:

1. Assist in preparing test cases and test data.
2. Execute tests assigned by senior testers or the test lead.
3. Learn testing tools, processes, and methodologies.
4. Perform basic functionality tests, smoke tests, and low-complexity tasks.
5. Support senior engineers in documentation and data preparation.
6. Help in maintaining test scripts and environment setup.
7. Observe and learn through participation in reviews and team meetings.

Why Pyramid Structure? (Exam Point)

1. Wider base = more testers responsible for hands-on execution.
2. Narrow top = few managers responsible for strategy and decisions.
3. Ensures smooth communication flow from top to bottom.
4. Provides clear responsibilities, accountability, and scalability.
5. Supports large projects where multiple resources need coordination.

Ready-to-Write 10–15 Marks Conclusion

The Testing Group Hierarchy ensures that the testing process is structured, disciplined, and effective.
A pyramid structure with **Test Manager → Test Leader → Test Engineers → Junior Test Engineers** provides clear control, competence development, and efficient workflow.

Definition of Software Metrics

A **software metric** is a *quantitative or countable measurement* of one or more properties of a software product, process or project. (BrowserStack)
In other words, metrics enable managers and engineers to **measure**, **monitor**, **control**, and **improve** software development, testing and maintenance. (Lansa)
One authoritative definition:

"Software metrics can be defined as *the continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products*. "

Key attributes of good software metrics:

- Quantitative (numerical) (Scaler)
- Understandable and well-defined measurement method (GeeksforGeeks)
- Repeatable/consistent when measured under the same conditions (Scaler)
- Applicable early and throughout development lifecycle (GeeksforGeeks)
- Economical to compute (cost vs benefit) (Scaler)

Types / Classification of Software Metrics

Multiple classifications exist. Commonly, metrics are grouped by **what they measure**: the *product*, the *process*, or the *project*. (Scaler)
Here are the main types:

1. Product Metrics

- Measure characteristics of the *software product* itself (code, design, architecture, documentation). (GeeksforGeeks)
- Examples: lines of code (LOC), cyclomatic complexity, size of modules, defect density, code coverage. (Wikipedia)
- Use: assess quality, maintainability, complexity, reliability of the software product.

2. Process Metrics

- Measure the *software development and maintenance process*. How well the process is working. (Scaler)
- Examples: effort variance (planned vs actual), schedule variance, defect injection rate, test case execution rate. (GeeksforGeeks)
- Use: improve the process, identify bottlenecks, monitor execution efficiency.

3. Project Metrics

- Focus on the *project management* aspects: cost, schedule, resources, productivity. (GeeksforGeeks)
- Examples: number of modules delivered per month, cost per function point, team productivity, defects per person-month.
- Use: monitor health of the project, support planning and decision-making.

Additional / Alternative Classifications

Some sources list further subdivisions, such as:

- Internal metrics (measuring inside the code or design) vs external metrics (measuring external quality from user's perspective) (Scribd)
- Direct measurement vs indirect measurement (directly measured attribute vs derived attribute) (GeeksforGeeks)
- Metrics oriented toward size, function, people, etc. (Wikipedia)

Summary Table

Metric Type	What it measures	Use case examples
Product Metrics	Software product properties (code, design, defects)	LOC, complexity, defect density
Process Metrics	How development/test/maintenance process is functioning	Effort variance, test cycle time, injection rate
Project Metrics	Project management aspects (cost, schedule, resources)	Productivity, cost per unit, modules delivered