UNIT III – PERFORMANCE AND REGRESSION

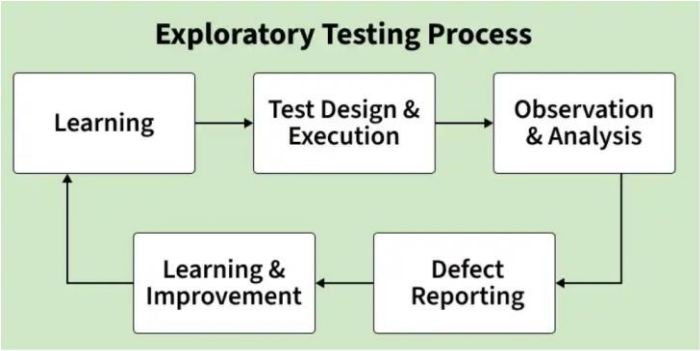
Factors – Methodology – Tools – Challenges. Regression Testing: Types – Methods. Internationalization Testing – Ad-hoc Testing: Buddy and Pair Testing – Exploratory Testing– Iterative Testing – Agile and Extreme Testing. Usability and Accessibility Testing.

Exploratory Testing

Exploratory Testing is a software testing technique in which testers examine the application without following predefined test cases. Instead, they learn about the system, design tests, and execute them simultaneously. It is an effective approach for identifying hidden defects and gaining a better understanding of how the application works.

- Testers do not rely on predefined test cases; they use their experience, skills, and intuition to perform testing.
- It is useful for uncovering unexpected defects, unusual scenarios, and edge-case issues.
- This method is flexible and works well for new features, early development stages, or applications that change frequently.

Exploratory Testing Process



Exploratory Testing Process

Exploratory Testing is a flexible testing process where the tester learns about the application, creates tests, and executes them at the same time. The tester explores different parts of the application to find defects and understand how it works**.**

Learning:
The tester explores the application to understand its features, functions, and possible problem areas.
Test Design & Execution:
The tester creates and performs test cases at the same time without using predefined test cases.
Observation & Analysis:
The tester watches how the application behaves and checks for errors, unexpected results, or defects.
Defect Reporting:
Any defects found are recorded and reported with enough details so developers can fix them.
Learning & Improvement:

The tester uses the experience gained to improve future testing and continue enhancing the software quality.

Types of Exploratory Testing

- **Freestyle Exploratory Testing:**
  The tester explores the application freely without a specific plan or test cases.It helps in understanding the application and finding unexpected defects.
- **Scenario-Based Exploratory Testing:**
  The tester checks the application using real-life user scenarios or workflows. It ensures the application works correctly in practical situations.
- **Strategy-Based Exploratory Testing:**
  The tester follows a specific testing approach, such as risk-based or boundary testing, to focus on important areas of the application.
- **Session-Based Exploratory Testing:**
  Testing is performed within a fixed time limit using a clear testing objective (test charter). It helps organize and track testing activities.
- **Pair Exploratory Testing:**
  Two people, such as a tester and a developer, work together to explore the application. This improves collaboration and helps find more defects.
- **Risk-Based Exploratory Testing:**
  The tester concentrates on the most critical and high-risk features of the application to identify defects that could have a major impact.

Advantages of Exploratory Testing

Exploratory Testing provides several benefits that help improve software quality and identify defects quickly.

- **Less Preparation:** Does not require detailed test plans or predefined test cases.
- **Finds Critical Defects Quickly:** Helps identify major defects at an early stage.
- **Increases Productivity:** Testers can use their knowledge and experience to test more effectively.
- **Encourages Creativity:** Allows testers to think creatively and explore different testing scenarios.

Iterative Testing

Iterative Testing is a software testing approach in which small, incremental changes are made to a product based on the results of previous tests and user feedback. After each change, the product is tested again to verify improvements, identify new issues, and ensure it meets predefined requirements. This continuous cycle of testing and improvement helps enhance software quality while making changes easier to manage and validate.

Benefits of Iterative Testing

- **Early Defect Detection:** Identifies and fixes problems at an early stage, reducing the chance of bigger issues later.
- **Early Working Prototype:** Helps create a functional version of the product early in the development process.
- **Easy Progress Tracking:** Each testing cycle has clear goals, making it easier to measure improvements.
- **Better Usability:** Finds usability issues and improves the overall user experience.
- **Supports Incremental Changes:** Allows small, manageable updates instead of large changes, reducing development risk.
- **Meets User Needs:** Continuous testing and feedback help build a product that better satisfies user requirements.

Steps of Iterative Testing

1. **Set the Goal:** Define a clear objective for the current testing iteration.
2. **Collect User Feedback:** Gather feedback from users to identify issues and improvement areas.
3. **Analyze Results:** Review the test results and decide what changes are needed.
4. **Retest and Improve:** Apply the changes, test again, and repeat the process for continuous improvement.

Agile and Extreme Testing:

Agile Testing is a software testing approach that follows agile methodologies, where testing is integrated with development activities such as requirement analysis, design, and coding. It is an iterative and continuous process performed throughout the Software Development Life Cycle (SDLC), emphasizing teamwork, regular feedback, continuous improvement, and customer satisfaction.

- Testing is carried out continuously in every iteration, helping to identify defects early and support quicker software releases.
- Developers, testers, and other team members work together closely through effective communication and collaboration.
- Regular customer feedback and a flexible approach help improve software quality and ensure the product meets user expectations.

Principles of Agile Testing

Agile testing integrates testing with the software development process to provide continuous feedback, quickly resolve defects, and ensure the product meets customer expectations. The key principles of Agile testing include:

- **Continuous feedback:** Frequent feedback during each iteration helps identify issues early, reducing both time and cost.
- **Testing alongside development:** Testing is performed simultaneously with development activities rather than waiting until coding is complete.
- **Collaborative teamwork:** Developers, testers, and business experts work together throughout the project.

- **Minimal documentation:** Agile relies on simple and essential documentation instead of extensive paperwork.
- **Quality-focused coding:** Defects are identified and corrected within the same iteration to maintain clean and reliable code.
- **Regular communication:** Ongoing feedback from stakeholders helps ensure the product satisfies business requirements.
- **Customer involvement:** Customers actively provide feedback and can request requirement changes at any stage of development.
- **Test-driven development:** Testing plays a key role in guiding development, resulting in faster delivery and higher software quality.

Agile Testing Methodologies

Agile testing methodologies emphasize flexibility, teamwork, and continuous improvement throughout the software development process. Some of the common Agile testing methods are:

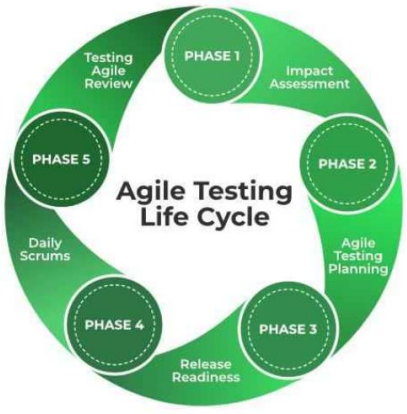
- **Test-Driven Development (TDD):** Test cases are created before writing the code, then the code is developed and improved through refactoring.
- **Behavior-Driven Development (BDD):** Development is based on the expected behavior of the software, with close collaboration among developers, testers, and business stakeholders.
- **Exploratory Testing:** Testers actively explore the application without predefined test cases to discover unexpected defects.
- **Acceptance Test-Driven Development (ATDD):** Acceptance criteria and tests are defined by developers, testers, and customers before implementation begins.
- **Extreme Programming (XP):** An Agile methodology that emphasizes simple design, continuous customer feedback, pair programming, and frequent software releases.
- **Session-Based Testing:** Testing is performed in planned, time-limited sessions, usually lasting between 45 and 90 minutes, with specific objectives.
- **Dynamic Software Development Method (DSDM):** An Agile framework that delivers projects through iterative development with active user participation and continuous testing.
- **Crystal Method:** An Agile approach that gives priority to people, communication, and adaptability rather than rigid processes and documentation.

Phases of Agile Testing



- **Iteration 0:** Initial setup phase where requirements are analyzed, risks are identified, resources are planned, and the testing environment is prepared.
- **Construction Iteration:** Core development phase where features are built and tested continuously using confirmatory testing (to verify requirements) and investigative testing (to find hidden defects).
- **Release Phase:** Final phase where system testing and acceptance testing are performed to ensure the product is ready for deployment.
- **Production:** The product is released to users, and any real-time issues are fixed through maintenance and updates.

Agile Testing Life Cycle



Life Cycle

The **Agile Testing Life Cycle** consists of several phases that support continuous testing, rapid feedback, effective collaboration, and the delivery of high-quality software during every sprint.

- **Impact Assessment:** Customer and stakeholder feedback is analyzed to understand requirements, identify necessary changes, and establish testing goals for the next iteration.

- **Agile Test Planning:** The team prepares the testing plan by defining the testing strategy, selecting tools, allocating resources, creating schedules, and identifying potential risks.
- **Release Readiness:** The completed features are thoroughly tested to verify they meet quality standards and are ready for release or require additional improvements.
- **Daily Scrums:** Brief daily meetings are held to review testing progress, discuss challenges, and coordinate the team's activities for the day.
- **Test Agility Review:** Periodic review meetings are conducted with stakeholders to assess testing results, collect feedback, and identify opportunities to improve future testing iterations.

Benefits of Agile Testing

Agile testing improves software quality through continuous testing, quick feedback, faster delivery, flexibility, and teamwork.

- **Early Bug Detection:** Finds defects early, reducing the time and cost to fix them.
- **Faster Delivery:** Testing in small iterations helps release software more quickly.
- **Better Software Quality:** Continuous testing and feedback reduce defects and improve reliability.
- **Higher Customer Satisfaction:** Regular customer feedback ensures the software meets user needs.
- **Flexibility and Teamwork:** Adapts easily to changing requirements through close collaboration between developers, testers, and stakeholders.

Extreme Testing

Extreme Testing is a software testing approach that pushes a system beyond normal operating conditions to identify its breaking points, improve reliability, and ensure the software works correctly under extreme situations.

- Tests the system under unusual or extreme conditions.
- Helps identify hidden defects and performance issues.
- Improves software reliability, stability, and quality.

Types of Extreme Testing

1. Extreme Testing in Agile/XP

This approach is used in **Agile** and **Extreme Programming (XP)**, where testing is performed continuously during development.

- Uses continuous testing throughout development.
- Includes unit, integration, and acceptance testing.
- Supports Test-Driven Development (TDD) and Behavior-Driven Development (BDD).
- Ensures quick feedback and faster software delivery.

2. Stress Testing and Chaos Testing

These techniques evaluate how the software behaves under extreme conditions.

Stress Testing

- Tests the software beyond normal workload.
- Checks system stability and performance under heavy load.
- Identifies the system's breaking point.

Chaos Testing

- Intentionally introduces failures such as server crashes or network issues.
- Verifies whether the system can recover from failures.
- Improves system reliability and fault tolerance.

3. Fuzz Testing

Fuzz testing checks how the software handles unexpected or invalid inputs.

- Provides random or incorrect input data.
- Detects crashes, security vulnerabilities, and unexpected behavior.
- Improves application security and reliability.

4. HALT and HASS Testing

These techniques are mainly used for **hardware and embedded systems**.

HALT (Highly Accelerated Life Testing)

- Tests prototypes under extreme temperature, vibration, and stress.
- Identifies design weaknesses before production.

HASS (Highly Accelerated Stress Screening)

- Tests finished products during manufacturing.
- Detects production and assembly defects.

Benefits of Extreme Testing

- Finds defects that normal testing may miss.
- Improves software reliability and stability.
- Enhances system performance under extreme conditions.
- Reduces the risk of failures after deployment.
- Helps deliver high-quality and robust software.

Applications of Extreme Testing

- Agile and Extreme Programming (XP) projects.

- Web and mobile applications.
- Cloud and distributed systems.
- Security testing of applications.
- Embedded systems and electronic devices.