GYAAN SANDHI

YOUR BRIDGE TO BRILLIANCE

UNIT 6

Testing Framework

Contents

Testing Framework:

- Software Quality
- Software Quality Dilemma
- Achieving Software Quality
- Software Quality Assurance Elements of SQA
- SQA Tasks, Goals and Metrics
- Formal Approaches to SQA
- Statistical Software Quality Assurance
- Six Sigma for Software Engineering
- ISO 9000 Quality Standards
- SQA Plan,
- Total Quality Management
- Product Quality Metrics
- In process Quality Metrics
- Software maintenance
- Ishikawa's 7 basic tools
- Flow Chart, Checklists, Pareto diagrams, Histogram, Run Charts, Scatter diagrams, Control chart, Cause Effect diagram.
- Defect Removal Effectiveness and Process.

Testing Framework

A Testing Framework is a set of rules, guidelines, tools, and best practices used to design, execute, and manage test cases in a structured way.

Why Testing Framework is needed

- Makes testing systematic and organized
- Improves reusability of test scripts
- Reduces maintenance effort
- Generates clear test reports

Key components of a Testing Framework

- Test Scripts – Automated or manual test cases
- Test Data – Input values used for testing
- Libraries – Reusable functions or methods
- Test Runner – Executes test cases
- Reporting – Shows pass/fail results

Software Quality Assurance (SQA)

Software Quality refers to how well a software product meets user requirements, performs reliably, and works without defects.

Characteristics of Software Quality

- Correctness – Does what it is supposed to do
- Reliability – Works consistently without failure
- Usability – Easy to use and understand
- Efficiency – Uses resources effectively
- Maintainability – Easy to update or fix
- Security – Protects data and prevents unauthorized access

Software Quality Dilemma

The Software Quality Dilemma is the conflict between delivering software quickly and ensuring high quality.

Why this dilemma occurs

- Limited time
- Limited budget
- Pressure from clients
- Changing requirements

Achieving Software Quality

Achieving Software Quality means developing software that is correct, reliable, efficient, secure, and meets customer requirements.

How Software Quality is Achieved

1. Clear Requirements
   - Requirements should be complete, correct, and well-documented
   - Avoids misunderstandings and rework

2. Proper Software Design
   - Use standard design principles
   - Modular and well-structured design improves quality

3. Following Coding Standards
   - Use proper naming conventions
   - Write clean, readable code
   - Helps in maintenance and reduces errors

4. Software Testing
   - Detects defects before delivery
   - Includes:
     - Unit Testing
     - Integration Testing
     - System Testing
     - Acceptance Testing

5. Automation and Tools
   - Automated testing tools reduce manual effort
   - Improves accuracy and speed

6. Reviews and Inspections
   - Code reviews and design reviews
   - Helps find defects early

7. Continuous Improvement
   - Collect feedback
   - Fix issues and improve processes

Software Quality Assurance (SQA)

Software Quality Assurance (SQA) is a planned and systematic process to ensure that software development follows defined standards and procedures.

Elements of Software Quality Assurance (SQA)

1. Standards
   - Coding standards
   - Documentation standards
   - Design standards

2. Reviews and Audits
   - Formal examination of:
     - Requirements
     - Design
     - Code
   - Ensures compliance with standards

3. Testing
   - Verifies and validates software
   - Ensures software meets requirements

4. Quality Control Activities
   - Detect and remove defects
   - Includes testing and defect tracking

5. Documentation
   - Proper documentation of:
     - Requirements
     - Design
     - Test cases
     - User manuals

6. Training
   - Improves skills of developers and testers
   - Ensures correct use of tools and methods

7. Measurement and Metrics
   - Measures quality using:
     - Defect density
     - Test coverage
     - Failure rate

8. Change Management
   - Manages changes efficiently
   - Prevents quality issues due to uncontrolled changes

SQA Tasks

Quality Planning

- Prepare SQA plan for the project
- Define:
  - Quality standards
  - Procedures
  - Tools
  - Roles and responsibilities

Example: *Deciding coding standards and testing strategy before development.*

Process Definition and Monitoring

- Define development and testing processes
- Ensure teams follow approved processes

Example: *Ensuring Agile teams follow sprint reviews and retrospectives.*

Reviews and Audits

- Requirements
- Design
- Code

- Conduct formal reviews of:
- Requirements
- Design
- Test cases

Example: *Code inspection to detect logic errors.*

Verification and Validation Support

- Ensure the software:
  - Verification: Built correctly
  - Validation: Built the right product

Testing Oversight

- **Monitor** test planning and execution

- Monitor test planning and execution
- Ensure adequate test coverage
- Track defects

Example: Using Git to manage *code versions.*

Configuration Management

- Control changes to:
  - Source code
  - Documents
  - Test cases

Example: Using Git to manage *code versions.*

Defect Management

- Log defects
- Track defect status
- Analyze root causes

Goals of SQA

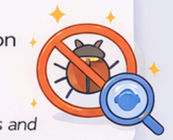
1 Ensure High Quality Software

- Deliver reliable, secure, and defect-free software

2 Prevent Defects

- Focus on early detection and prevention
- Reduce cost of fixing bugs later

Example: *Ensuring Agile teams follow sprint reviews and*



3 Ensure Process Compliance

- Ensure adherence to:
  - Organizational standards
  - Industry standards (ISO, CMMI)

Example: *Code inspection to detect logic errors.*

4 Improve Customer Satisfaction

- Meet user expectations
- Reduce failures after release

Example: *Reduce failures after release*

5 Reduce Development Cost and Time

- Early defect detection reduces rework
- Improves productivity

Example: Using Git to manage code versions.

6 Continuous Process Improvement

- Identify weaknesses
- Improve future projects

Example: More proces te change vernents.

SQA Metrics

SQA (Software Quality Assurance) metrics are quantitative measures used to assess the quality, effectiveness, and efficiency of software processes and products. They help the SQA team monitor progress, detect defects, and improve software quality during the software development life cycle (SDLC).

Product Metrics

- **Measure software product quality.**
- **Examples:**
  - **Number of defects per module**
  - **Code complexity (e.g., cyclomatic complexity)**
  - **Defect density (defects per KLOC – thousand lines of code)**

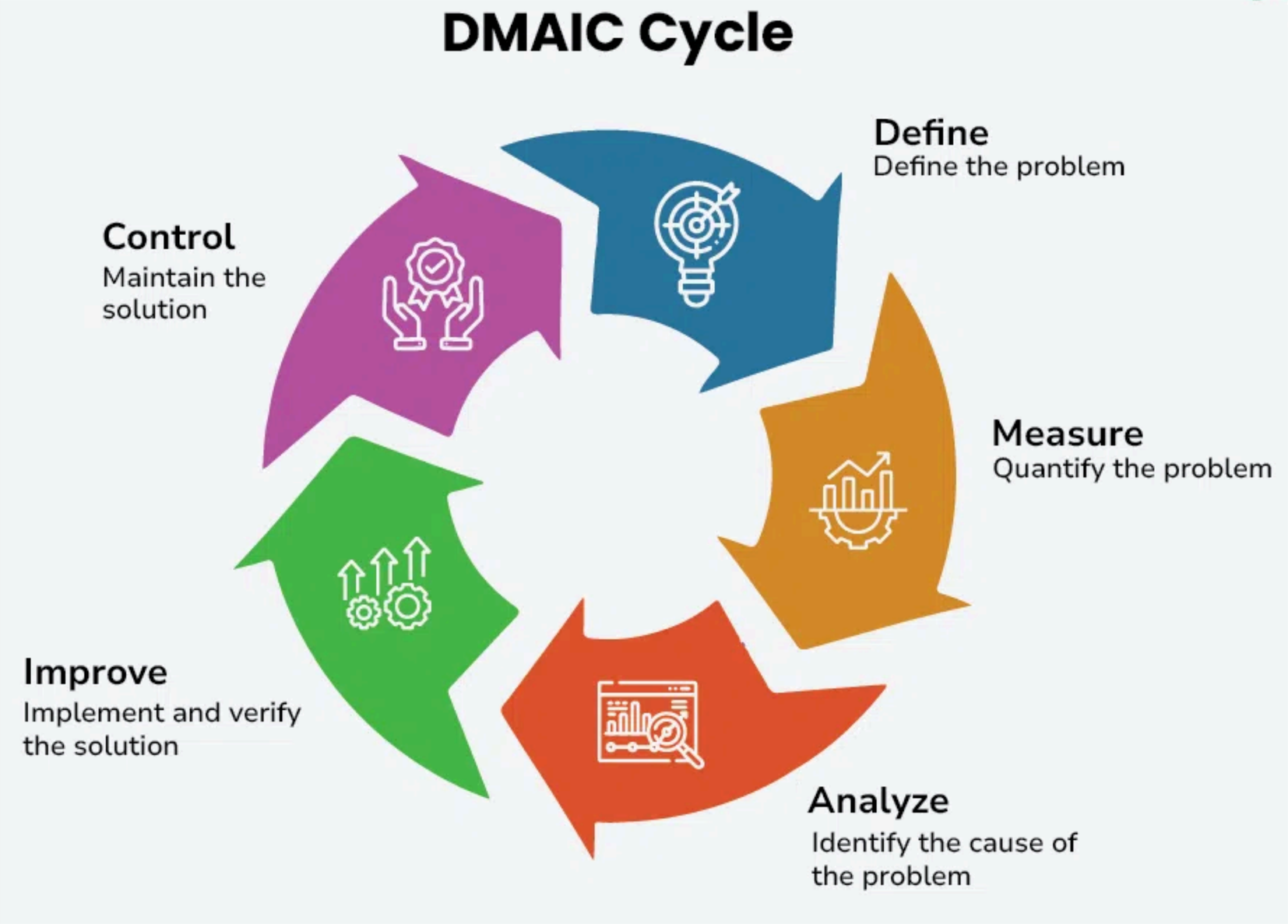
Process Metrics

- **Measure development process quality.**
- **Examples:**
  - **Number of test cases executed vs planned**
  - **Requirement review effectiveness**
  - **Percentage of defects found during early phases (prevention rate)**

Project Metrics

- **Measure project progress and performance.**
- **Examples:**
  - **Schedule adherence**
  - **Cost variance**
  - **Resource utilization**

Six Sigma


DMAIC Cycle
Define
Define the problem
Measure
Quantify the problem
Analyze
Identify the cause of the problem
Improve
Implement and verify the solution
Control
Maintain the solution

ISO 9000 Quality Standards

- **Customer focus** – All work should aim to satisfy customer needs and expectations.
- **Leadership** – Leaders provide clear vision and direction for the team.
- **Engagement of people** – Employee involvement and commitment improve quality.
- **Process approach** – Managing work through processes increases efficiency and consistency.
- **Improvement** – Continuous improvement and innovation strengthen the organization.
- **Evidence-based decision making** – Decisions based on data and facts reduce risk.
- **Relationship management** – Strong relationships with suppliers and stakeholders support growth.

SQA Plan

An SQA Plan (Software Quality Assurance Plan) is a formal document that defines how quality will be ensured throughout the software development life cycle.

Key Points:

- It describes quality goals, standards, and procedures for the project.
- Defines roles and responsibilities of developers, testers, and SQA team.
- Specifies review and audit activities.
- Includes testing strategy, tools, and techniques.
- Explains defect tracking, reporting, and corrective actions.

Importance:

- Ensures process compliance
- Prevents defects
- Improves overall software quality

Total Quality Management (TQM)

Total Quality Management (TQM) is a management approach that focuses on continuous improvement of products, processes, and services by involving everyone in the organization.

Key Points:

- Customer satisfaction is the main focus.
- Emphasizes continuous improvement.
- Involves all employees, not just QA team.
- Uses process-oriented approach.
- Focuses on defect prevention rather than detection.

Benefits:

- Improved software quality
- Reduced defects
- Higher customer satisfaction

Product Quality Metrics (After Development)

Measure the quality of the final software product from the user's point of view.

- Defect Density – defects per size of software
- Defect Leakage – defects found after release
- Reliability – failure rate / MTBF
- Performance – response time, resource usage
- Usability & Customer Satisfaction

In-Process Quality Metrics

Measure the quality during software development to prevent defects early.

- Defect Detection Rate per phase
- Defect Removal Effectiveness (DRE)
- Review Effectiveness
- Test Coverage (%)
- Rework Percentage
- Phase Containment Effectiveness

Key Difference:

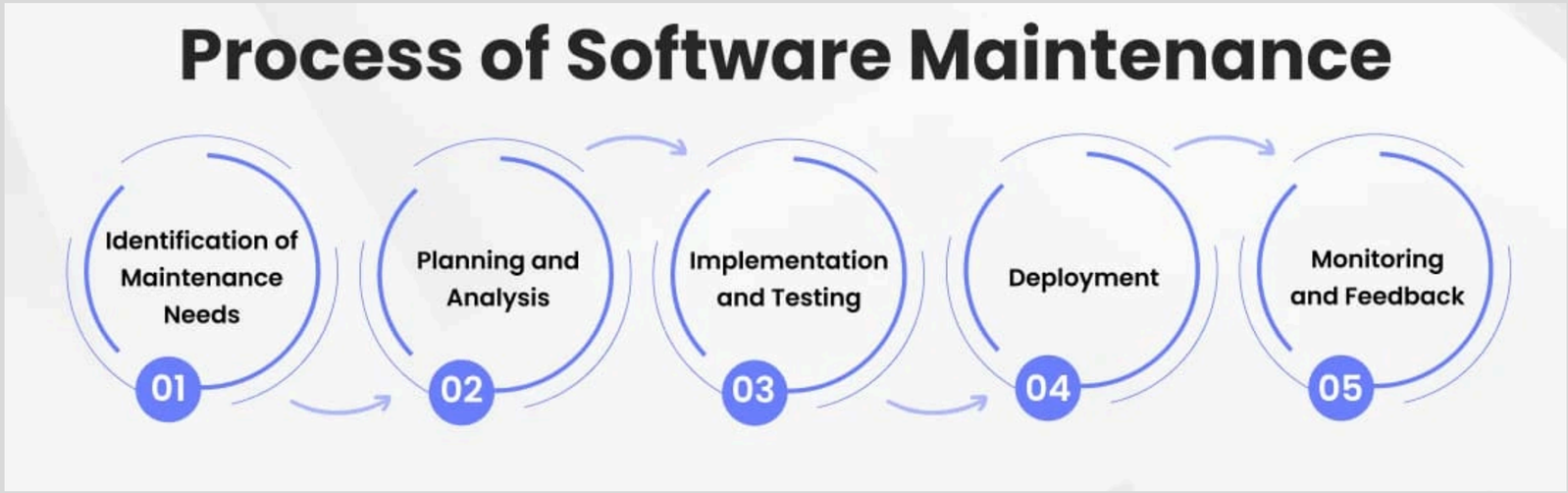
Product metrics evaluate what is delivered, while in-process metrics evaluate how it is built.

Software Maintenance

Software Maintenance is the process of modifying, updating, and improving software after it has been delivered to the customer.

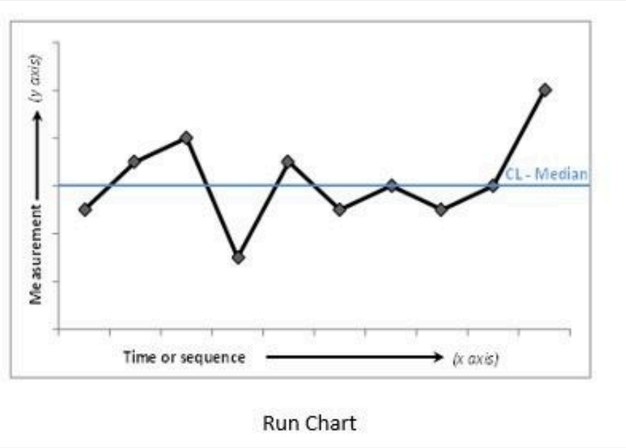
Maintenance is required to:

- Fix errors
- Improve performance
- Add new features
- Adapt software to changes

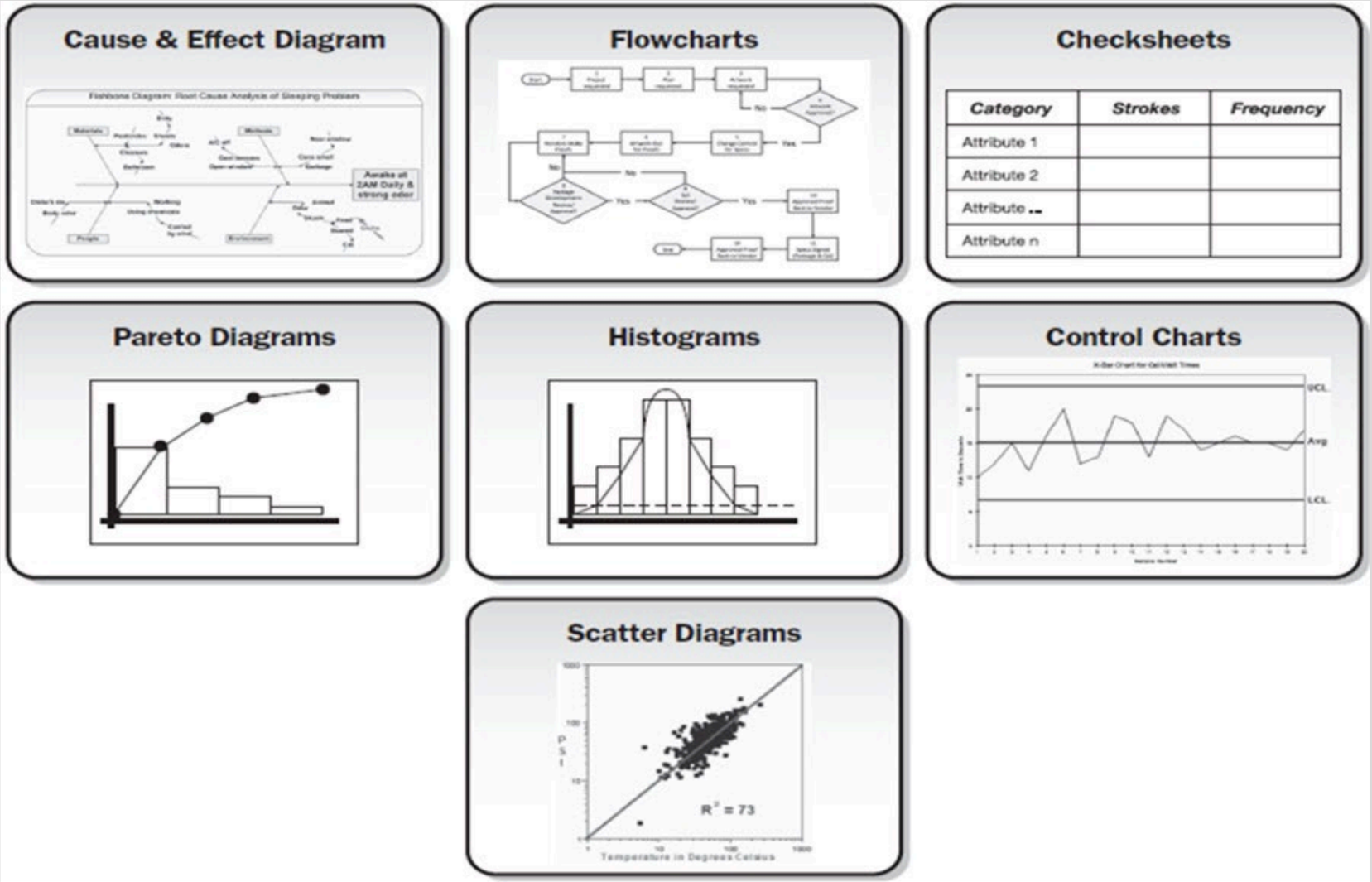


Ishikawa's 7 basic tools

- Flow Chart
- Checklists
- Pareto diagrams
- Histogram
- Run Charts
- Scatter diagrams
- Control chart
- Cause Effect diagram



Run Chart



Defect Removal Effectiveness and Process

Defect Removal Effectiveness (DRE) is a measure of how effectively defects are found and removed before the software is released.

Formula

$$\textbf{DRE} = \frac{\textbf{Defects removed before release}}{\textbf{Total defects (before + after release)}} \times 100$$

Example

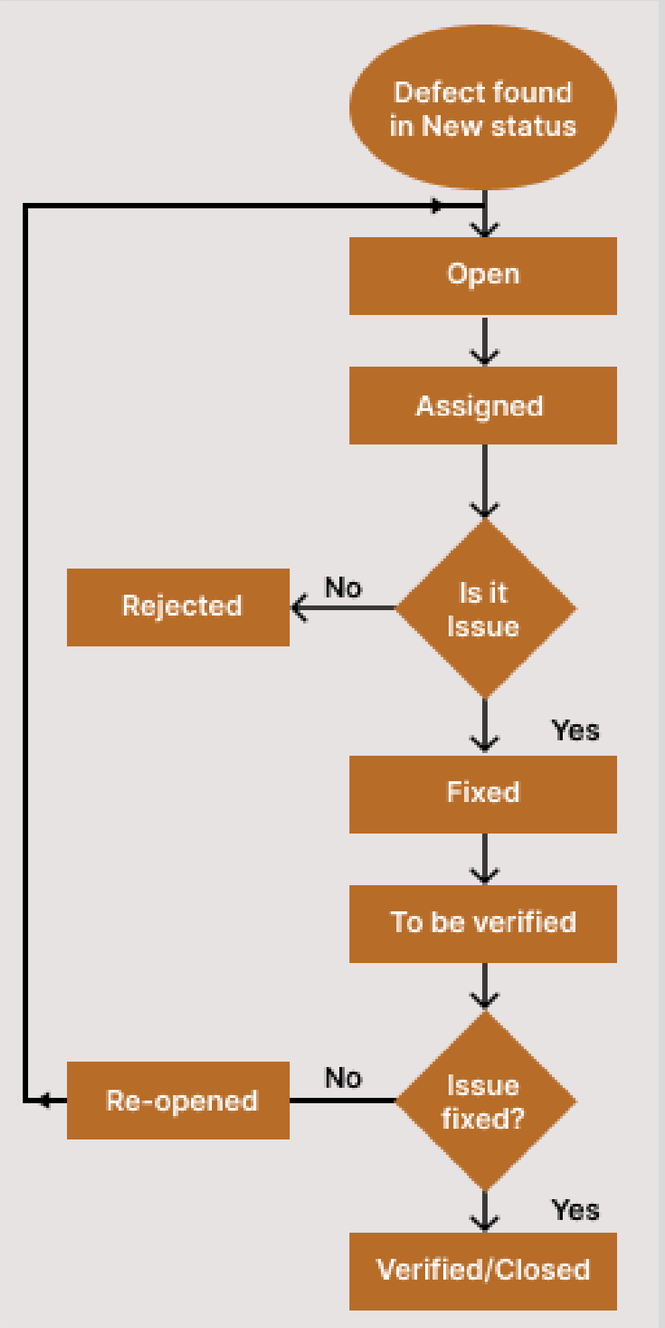
- Defects found before release = 90
- Defects found after release = 10

$$\text{DRE} = \frac{90}{90+10} \times 100 = 90\%$$

This means testing removed 90% defects before release, which is good quality.

Defect Removal Process

- Defect Found (New): Tester finds a new bug and reports it.
- Open: The bug is valid and accepted in the system.
- Assigned: The bug is given to a developer to fix.
- Is it Issue?: Check whether the bug is a real issue or not.
- Rejected: The bug is not valid or is a duplicate.
- Fixed: The developer has corrected the bug.
- To Be Verified: The tester re-tests the fixed bug.
- Issue Fixed?: Confirms whether the bug is resolved or not.
- Re-opened: The bug is opened again if it is not fixed.
- Verified/Closed: The bug is successfully fixed and closed.



Thank You