Brighter Blue

Basic Testing>Day 9

1. Defect Management

2. Regression Testing

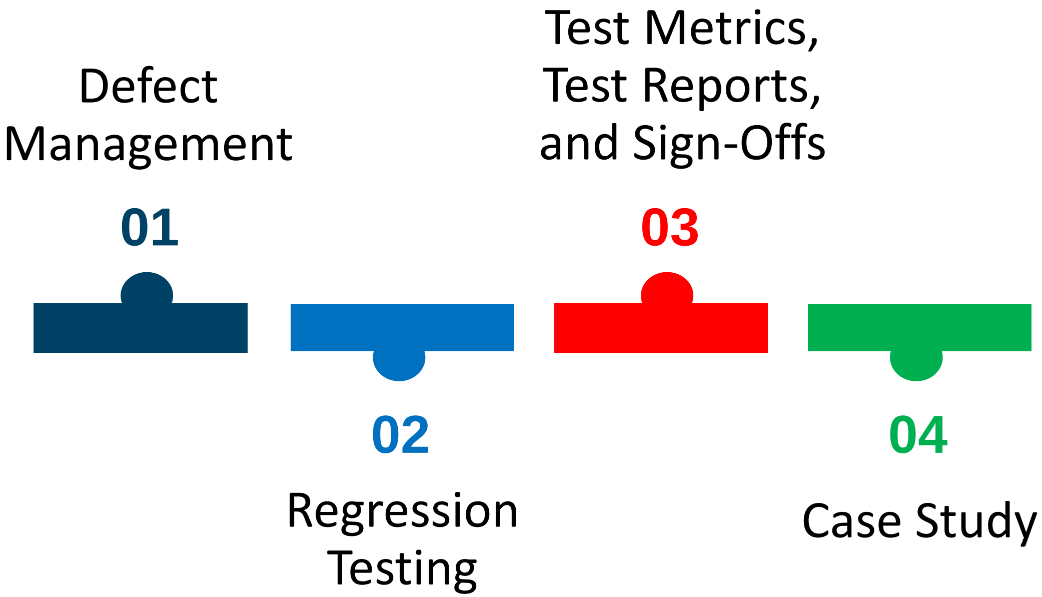
3. Test Metrics, Test Reports, and Sign-off

4. Case Study

IBM

Agenda

Day 09



Defect Management

01

Test Metrics, Test Reports, and Sign-Offs

03

02

Regression Testing

04

Case Study

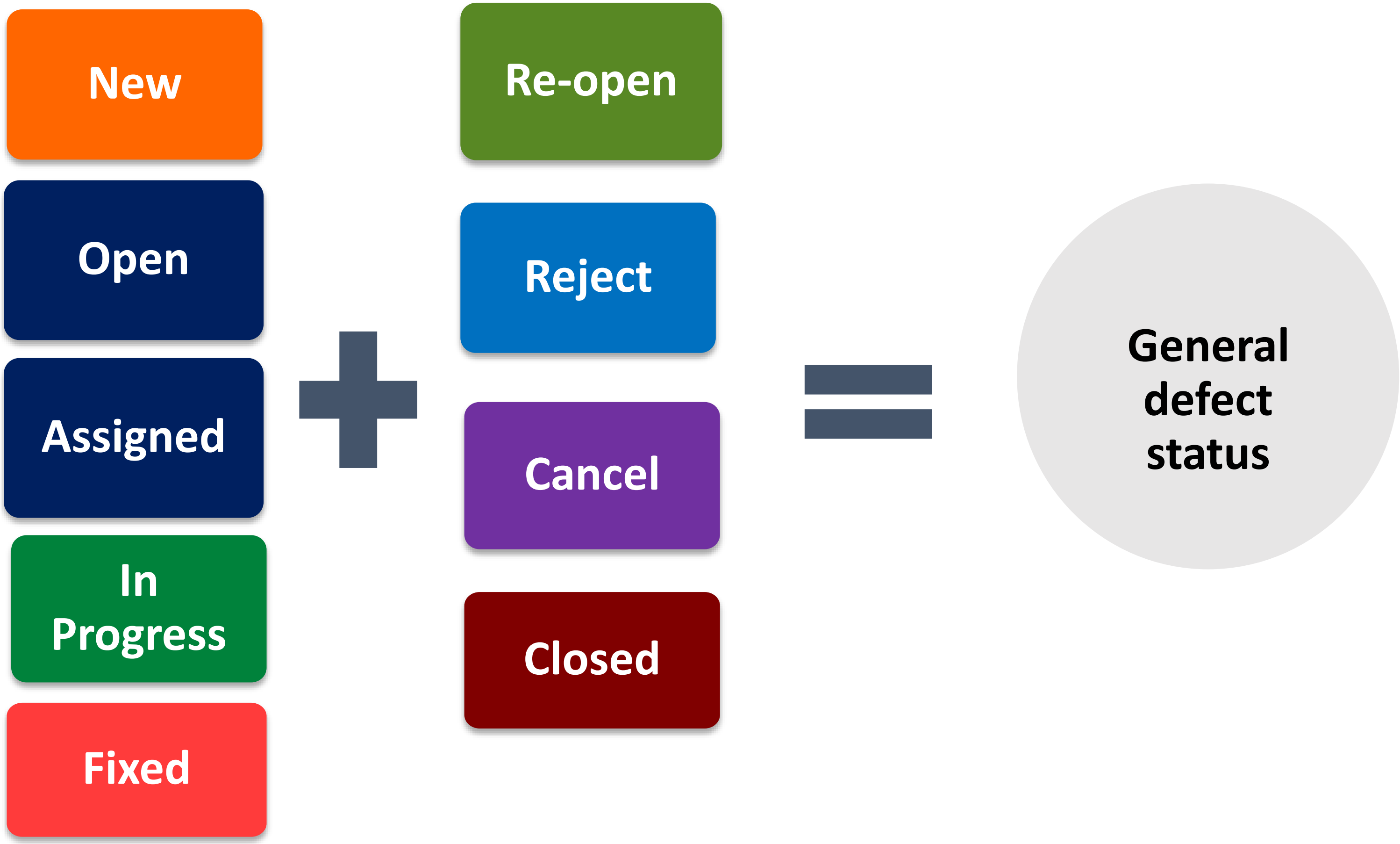
01 Defect Management

What is a defect?

A Software Defect / Bug is a condition in a software product which does not meet a software requirement (as stated in the requirement specifications) or end-user expectations (which may not be specified but are reasonable). In other words, a defect is an error in coding or logic that breaks and program fails to perform or malfunction or produces incorrect / unexpected results.

Defect **lifecycle**

The defect lifecycle covers the defect’s status when it is reported until successfully retested and Closed or Cancelled.



New

Open

Assigned

In Progress

Fixed

+

Re-open

Reject

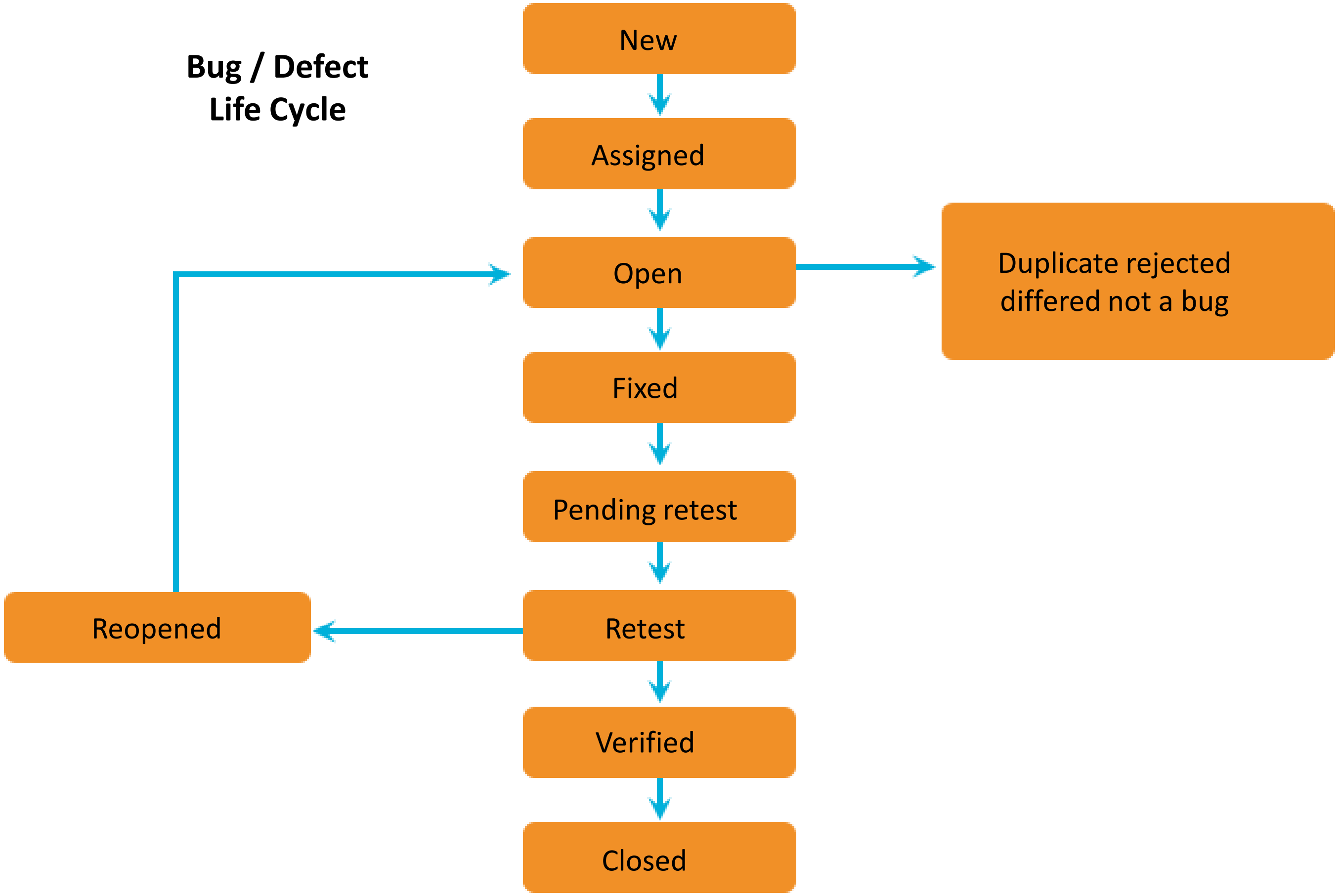
Cancel

Closed

=

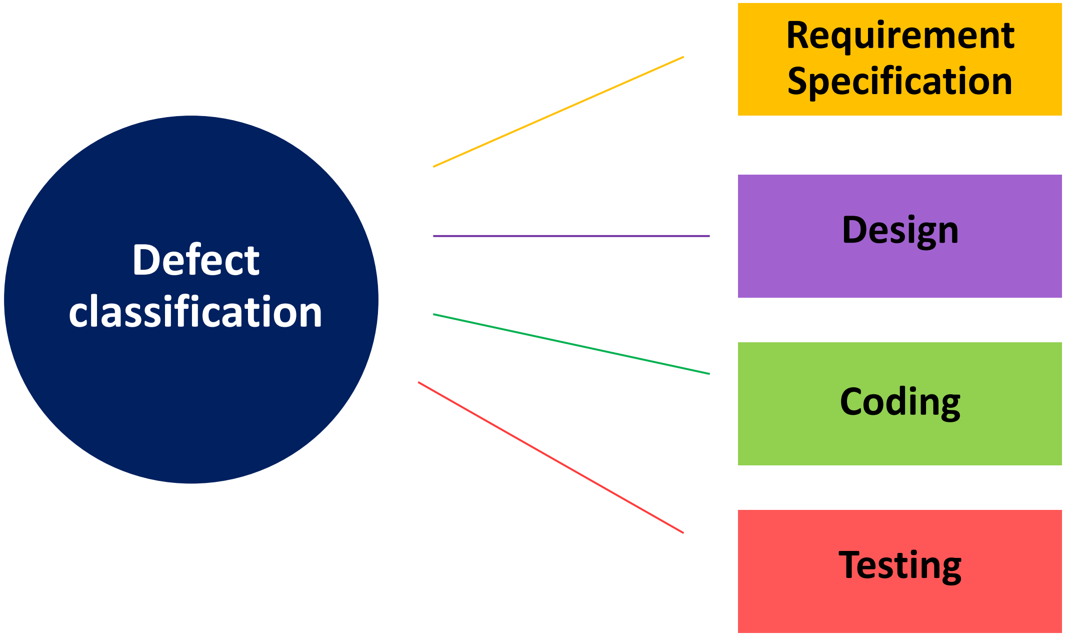
General defect status

Defect lifecycle **(continued)**



Defect **classes**

Defects can be classified in many ways. Organizations should adopt a single classification scheme and should apply it to all the projects. Defects can be classified under four major classes on the basis of their point of origin in the software life cycle.



Defect classification

Requirement Specification

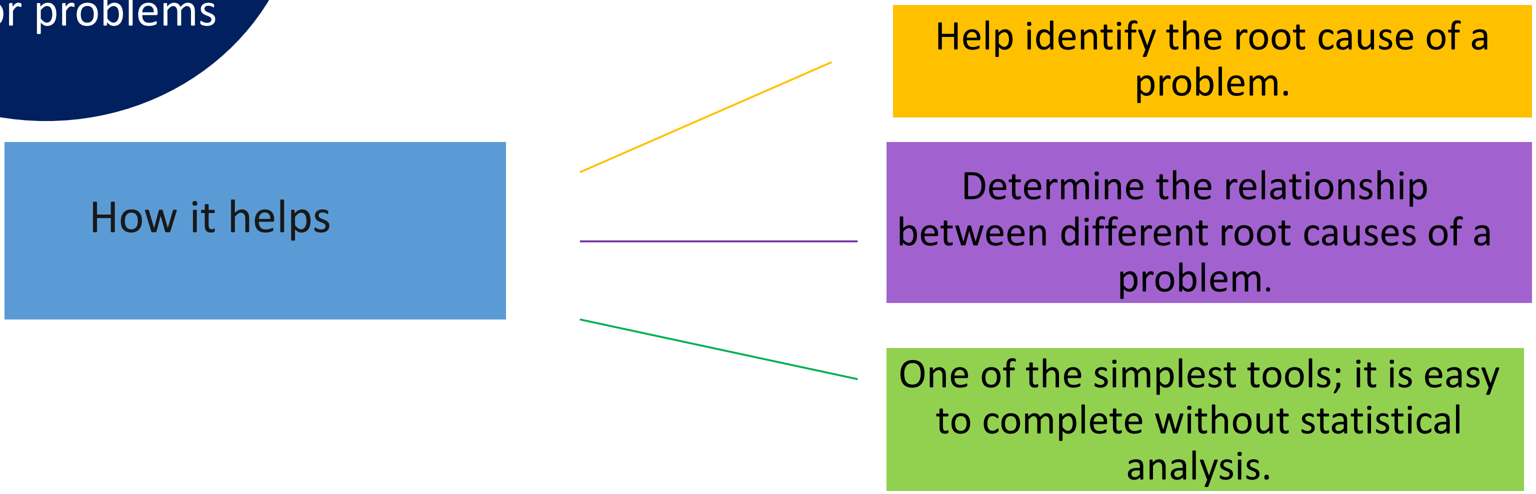
Design

Coding

Testing

Defect: root cause analysis

Root cause analysis (RCA) is a method of problem solving used for identifying the root causes of faults or problems



How it helps

Help identify the root cause of a problem.

Determine the relationship between different root causes of a problem.

One of the simplest tools; it is easy to complete without statistical analysis.

Requirement specification defects

Excessive functionality

Defects injected in the early phase can persist and can be very difficult to remove in the later phases.

Requirement documents are written in natural language which gives rise to ambiguity, contradiction, unclear, redundant, and imprecise requirements.

Requirement specification defects (continued)

What is the solution?

Organizations use a formal specification language accompanied by tools to prevent these defects.

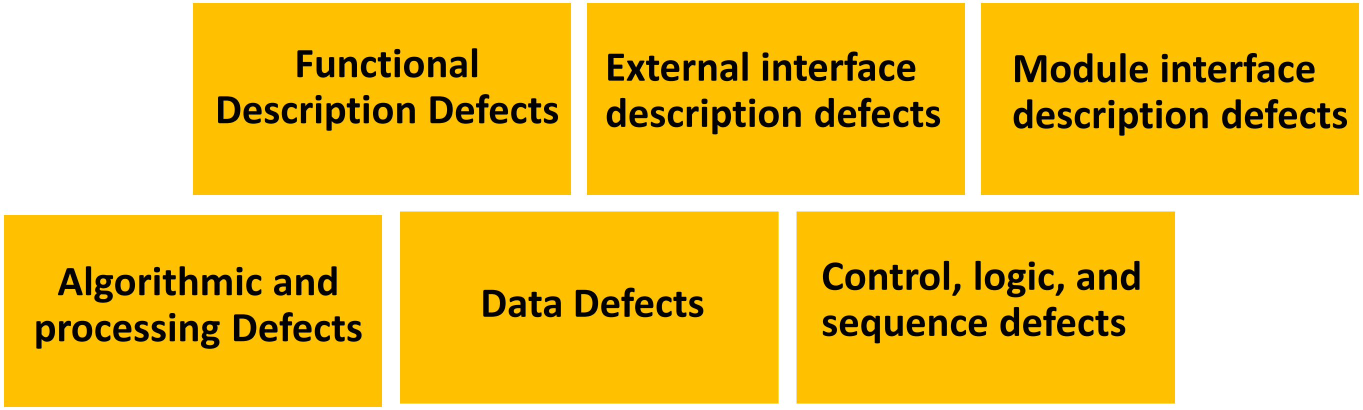
Requirement Defects

- Functional description defects
- Feature defects
- Feature interaction defects
- Interface description defects

Design defects

Design defects occur when system components, interactions between system components, interactions between the components outside software or hardware or users are incorrectly designed.

Types of Design Defects:



Functional Description Defects

External interface description defects

Module interface description defects

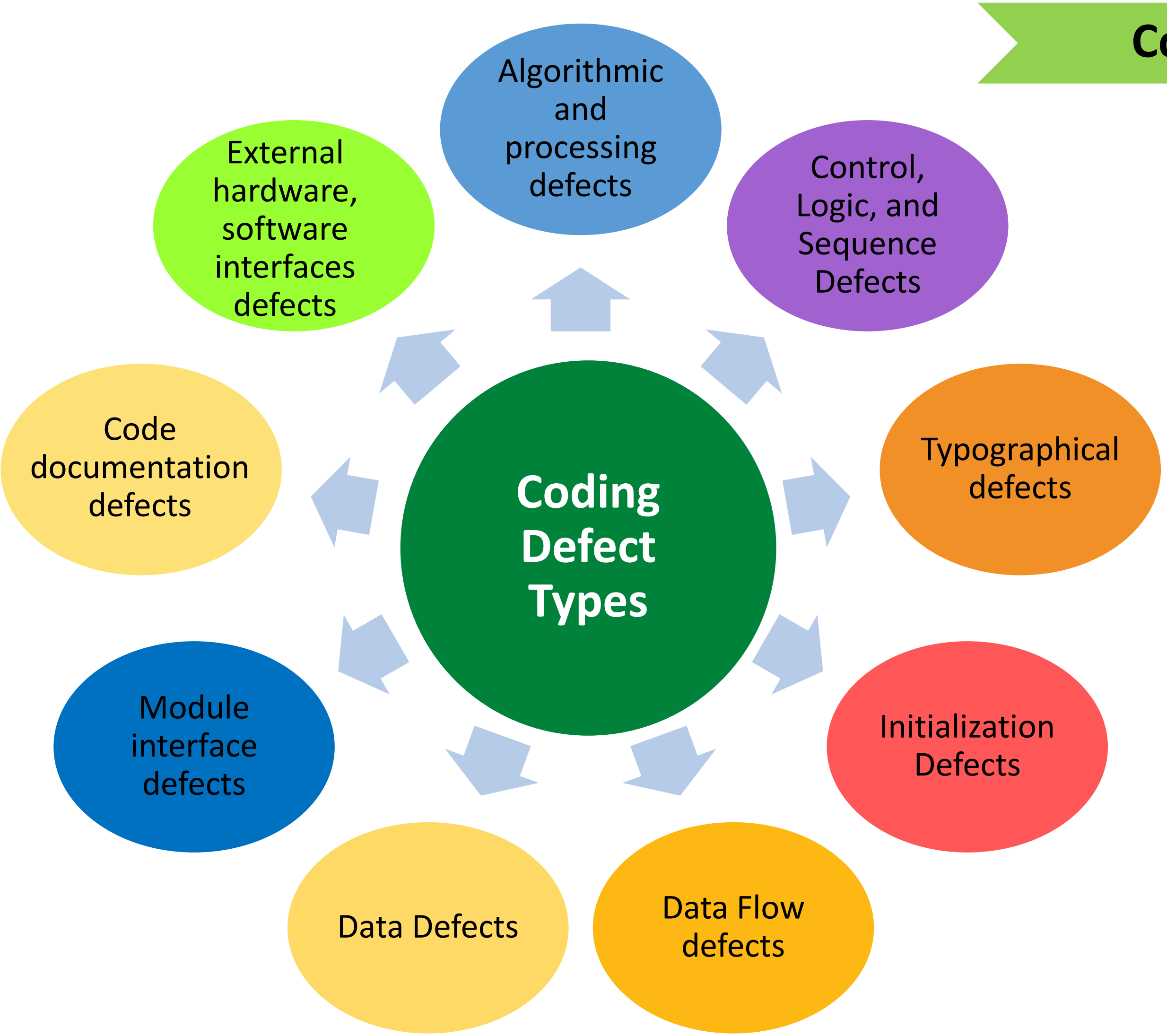
Algorithmic and processing Defects

Data Defects

Control, logic, and sequence defects

Coding defects

Coding



Coding Defect Types

Algorithmic and processing defects

Control, Logic, and Sequence Defects

Typographical defects

Initialization Defects

Data Flow defects

Data Defects

Module interface defects

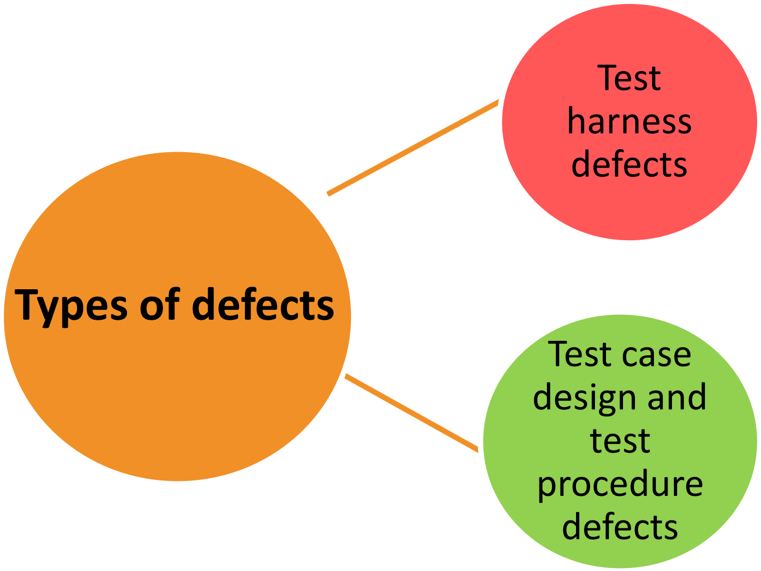
Code documentation defects

External hardware, software interfaces defects

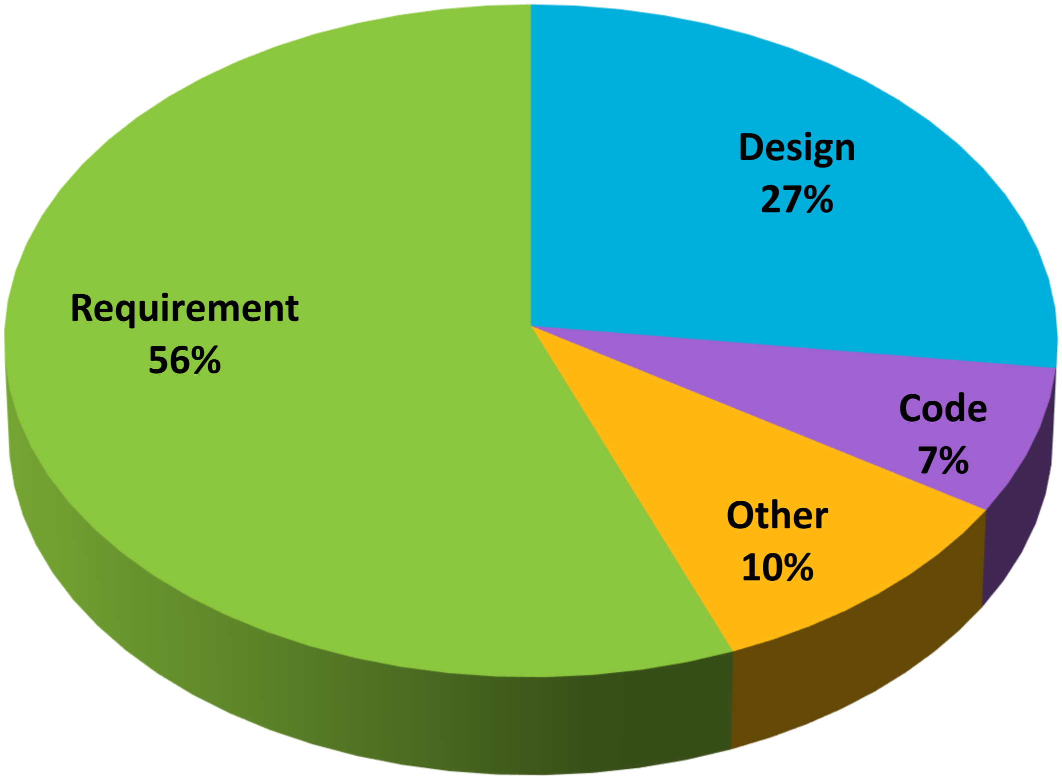
Testing defects

Causes of defect

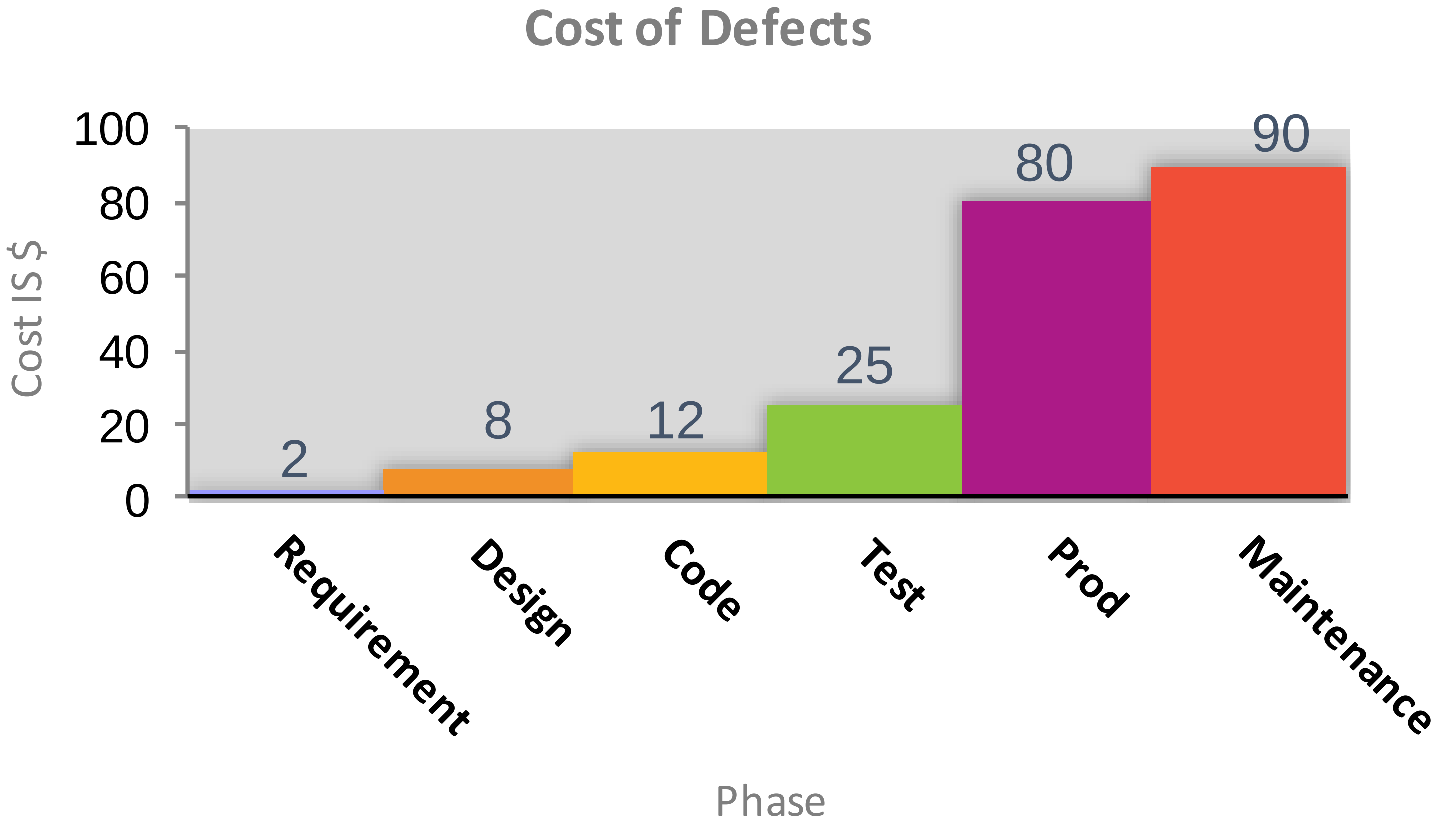
- Improper test cases
- Improper test process
- Improper test plan
- Improper test harness



Defect origination

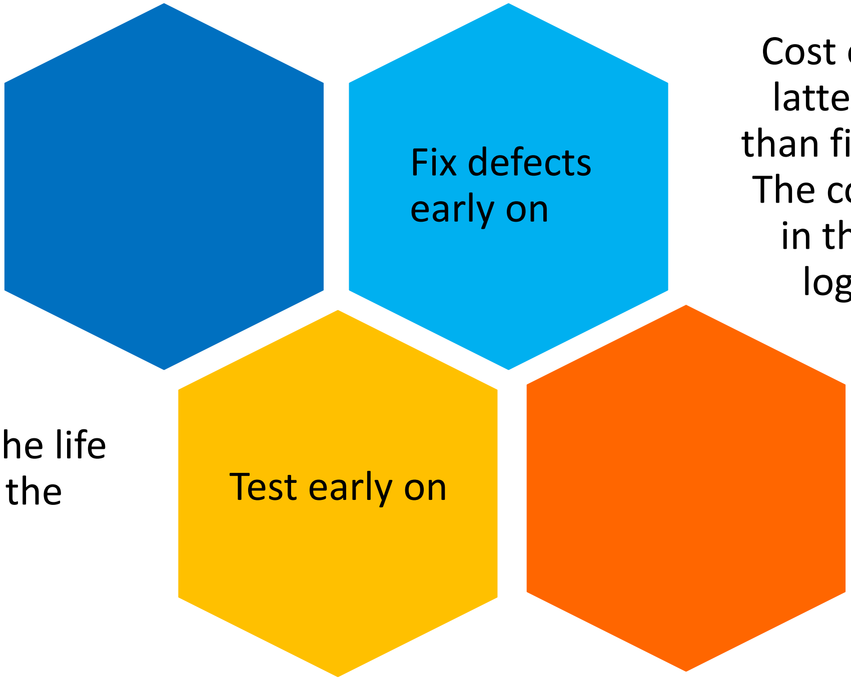


Relative cost of fixing defects



Prevention of defects

Ways to prevent defects



Cost of fixing the errors at a latter stage is much higher than fixing it at an early stage. The cost of rectifying defects in the latter stages climbs logarithmically and not linearly.

Testing early in the life cycle reduces the defects.

Cost of errors

A single error can cause death or injury if it fails in case of safety critical applications.

An AA jet crashed in Colombia (SA) because the captain entered an incorrect one-letter computer command that sent the jet into a mountain, killing 158 people aboard.

NYSE fined TD Waterhouse Investor Services US $225,000 for its website failures—inability to file online stock orders and inadequate customer service.

Legal consequences of defective testing

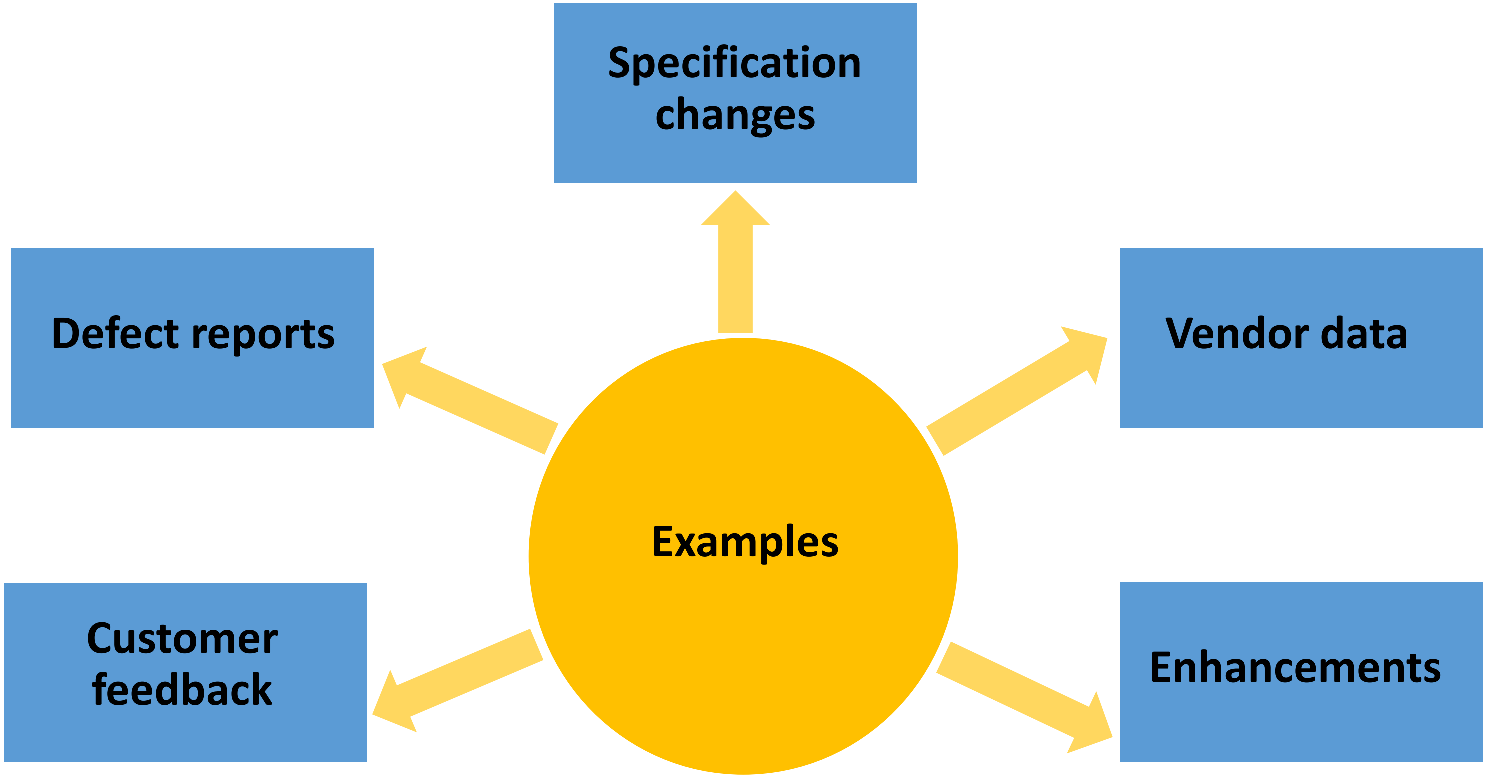
Test whether all **security procedures** are correctly and properly implemented.

Security testing attempts to verify that **protection mechanisms** built into a system will, in fact, protect it from improper penetration.

During **security testing**, password cracking, unauthorized entry into the software, network security are all taken into consideration.

Defect and change **tracking**

Change tracking is the process of tracking all the different kinds of requests to change a product.



Examples

Specification changes

Vendor data

Enhancements

Customer feedback

Defect reports

Log Change Requests (CRs)

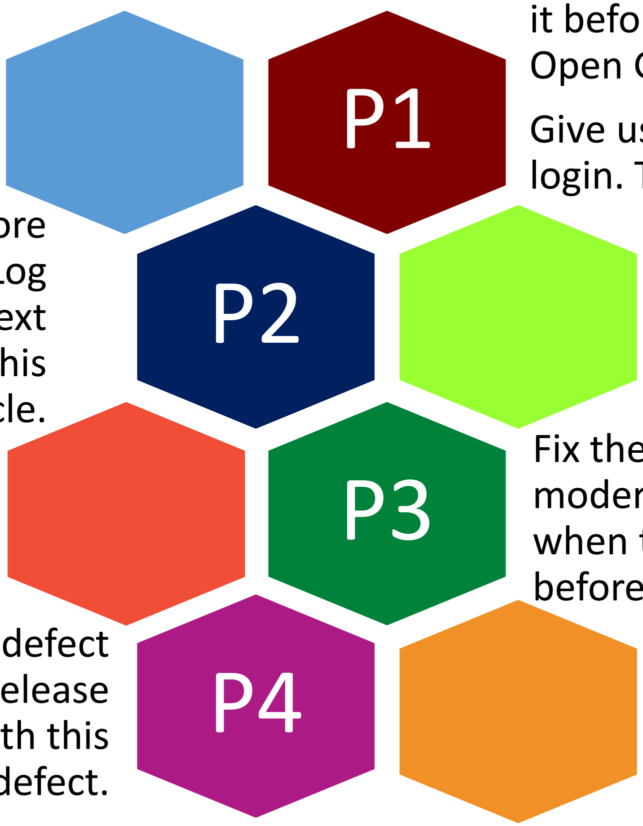
Use defect and change request tracking guidelines or standards:

- Document Change Requests
- Document the Defects as follows:
  - Describe how to reproduce; include test script if developers also use automated test tools.
  - Attach screen prints.
  - Put details of Environment (OS, browser and version, and so on).
  - Assign severity or criticality or priority.

Test defect **metrics**

Priority and severity

Defect priority provides a perspective for the order of the defect fixes. Priority, in other words tells us, how soon the defect has to be fixed. For example, the priority can be divided into P1, P2, P3, and P4.



Fix the defect on highest priority, fix it before the next build. Example: Open Gmail page

Give user name Password and click login. The page fails

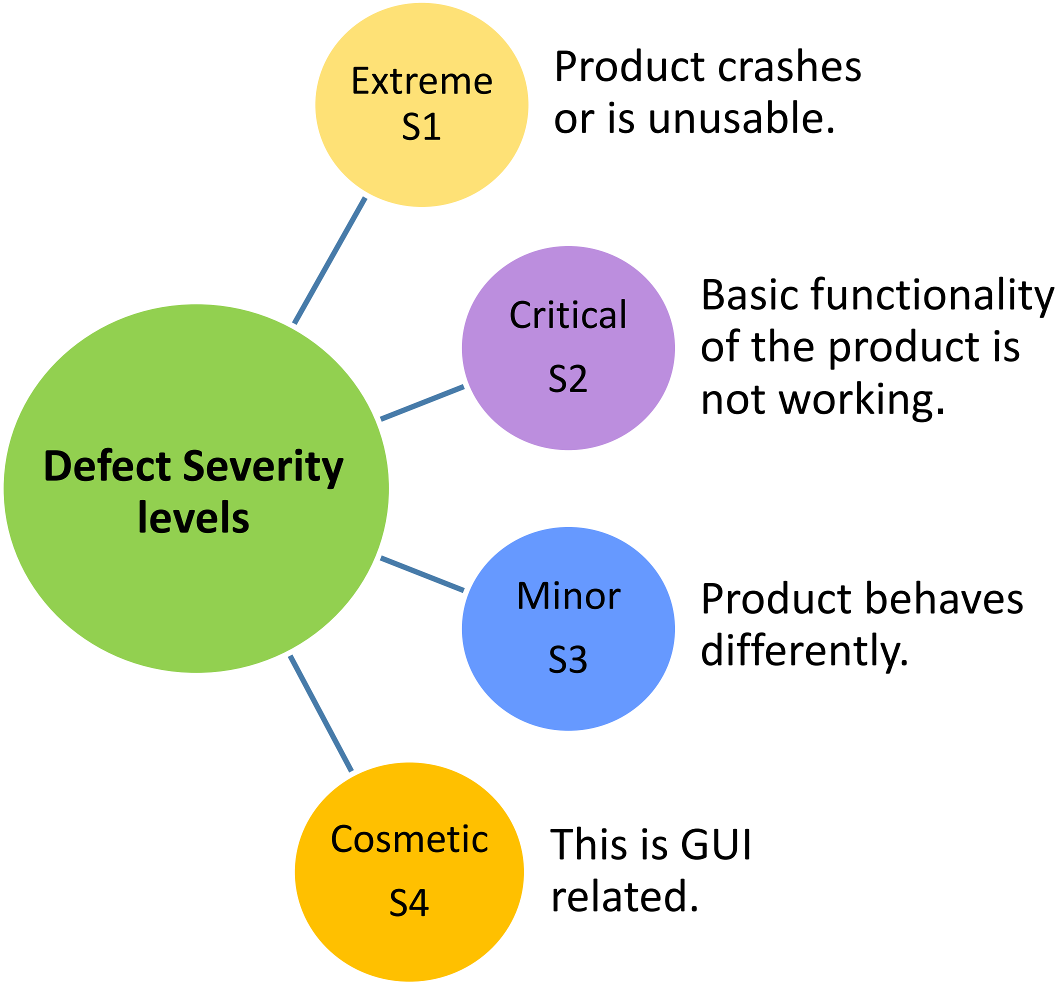
Fix the defect on high priority before the next test cycle. Example: Log into Gmail Page and Click on Next button. Button does not work so this can be fixed before next cycle.

Fix the defect on moderate priority when time permits, before the release.

Postpone the defect for the next release or live with this defect.

Defect **severity**

Defect Severity provides the perspective of the impact of that defect in product functionality.



Defect Severity levels

Extreme S1 — Product crashes or is unusable.

Critical S2 — Basic functionality of the product is not working.

Minor S3 — Product behaves differently.

Cosmetic S4 — This is GUI related.

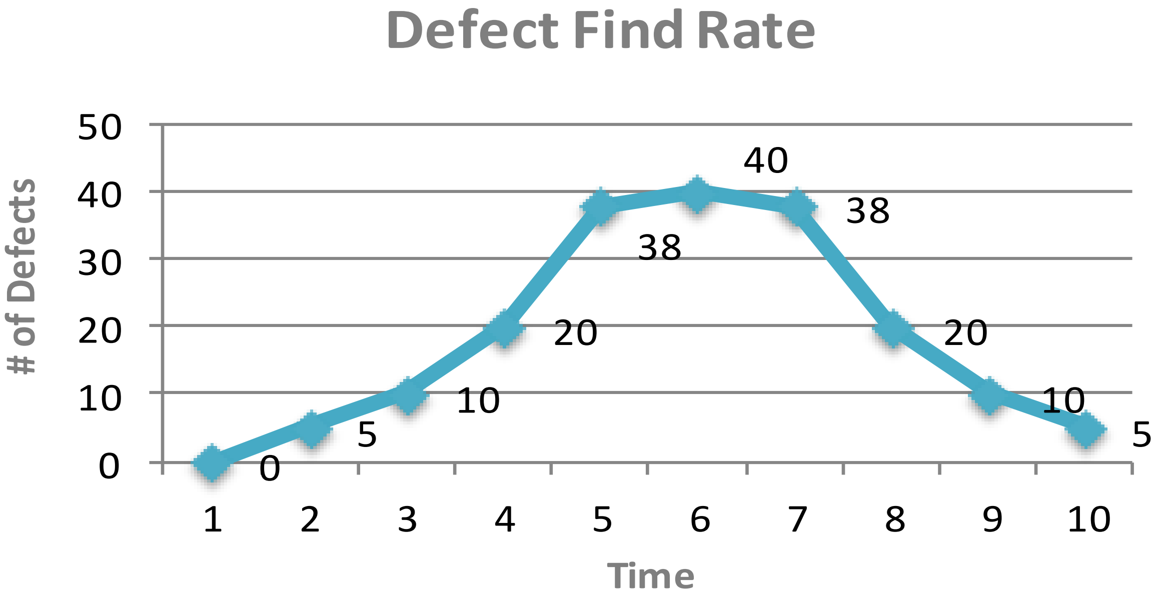
Defect **find and fix** rate

Find Rate

Defect find rate is a measure of tracking and plotting the total **number of defects found** in the product at regular intervals (say, daily, or weekly) from beginning to the end of the product development cycle.

Fix Rate

Defect fix rate is a measure of tracking and plotting the total **number of defects fixed** in the product at regular intervals (say, daily, or weekly) from beginning to the end of the product development cycle.



TIP:
The same bell curve also applies to fixed rate.

Other defect **metrics**

Outstanding defect rate

This is the number of outstanding defects which should be close to zero at all times in a well executed project during the entire test cycle.

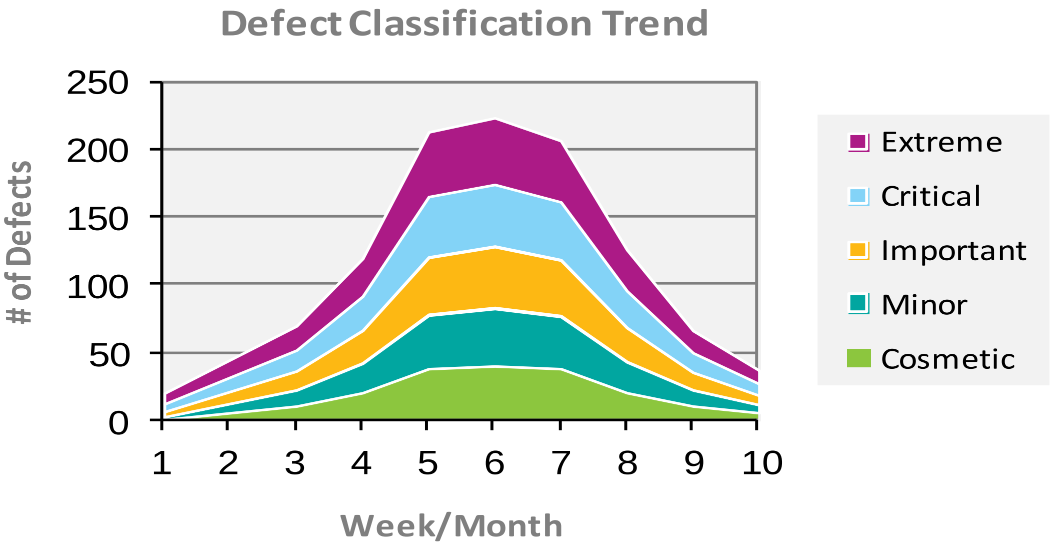
Priority outstanding rate

It provides additional focus on those defects that matter for the release.

Defect classification trend

It measures the product from a release perspective of the defect classification in the chart and helps in finding out the release readiness of the product:

- How many are extreme defects
- How many are critical
- How many are important



Common defect tracking **tools**

BugZilla: Available for free download at www.bugzilla.com

Test Track Pro: From Sea Pine

IBM Rational clear quest

Track Gear: from Logi Gear Corporation

PR Tracker: www.prtracker.com

HP Quality Center: Defect module (Licensed tool)

Questions?

Spot Quiz

01

Which of the following cannot be the Requirement Defect?

A Functional Description Defect

B Algorithmic Defects

C Future Defect

D Interface Description Defect

Spot Quiz

02

To achieve quality (that is, defect free products and services), we require:

A Close cooperation between management and staff

B Commitment

C An environment in which quality can flourish

D All of the above

Spot Quiz

03

Defect is any variance between actual and _________results.

A Expected

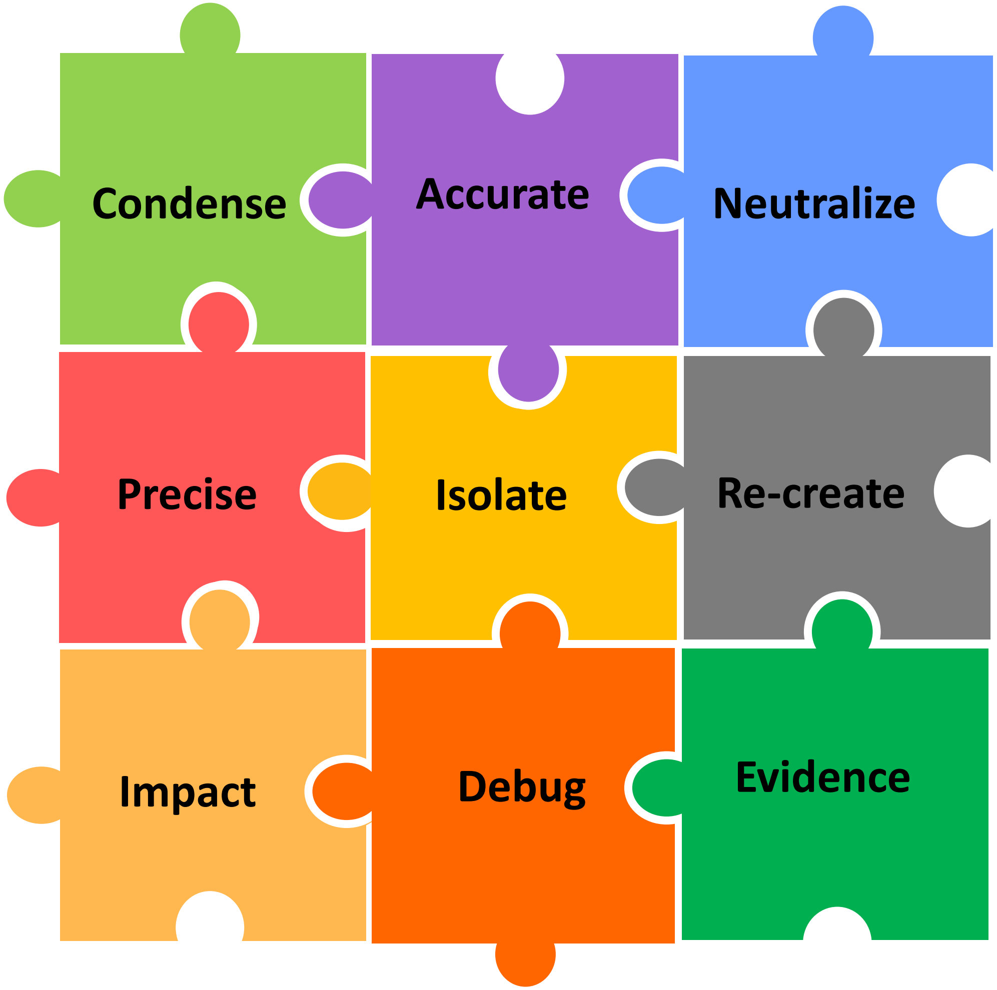
B True

C Perfect

D Wrong

Defect **remarks**

Here are some key points to make sure the next defect report you write is an effective one.



Use **checklist** before entering a defect

- Have you reproduced the defect again before entering the defect?
- Have you checked for duplicate defects in the same or similar kind of module?
- Verify that synopsis entered is clear and concise, and should contain keywords that you would relate with this. This makes it easier to search for later. If your synopsis goes over the width of the field, it is too long.
- Have you selected type as “defect”?
- Have you selected the relevant values from the following for your project:
  - Project Name
  - Release Name
  - OS Group
  - Version
  - HW Platform
  - Functionality
  - Feature Group and area

Use **checklist** before entering a defect (continued)

- Have you selected the most appropriate severity of the defect based on the defect complexity?
- Have you entered the description for:
  - Expected behavior
  - Observed behavior
- Have you entered the description for steps to reproduce to contain detailed reproductive steps so that reviewers or developers will not request for more information?
- What are the builds used for testing?
- What about connection information like database, server name, and so on?
- Have you attached screenshot(s), log file(s), trace file(s), sample report(s), Sample application(s) for defect?
- Have you verified that there are no typo errors within adapt entry?

Avoiding **duplication**

Avoiding Duplication

Duplication of bugs is a major hurdle for the testing community which results in the wastage of:

- Effort
- Time and
- Money

This is because more than one resource may report the same defect or the same resource may raise the same defect more than once.

This not only results in wastage of time among the testers, but also the developers.

The goal is to provide a methodology on how to minimize the duplication of bugs / defects which are reported in the software world; thereby saving time and money to the organization.

Questions?

Spot Quiz

01

Faults found by users are due to:

A Poor quality software

B Poor software and poor testing

C Bad luck

D Insufficient time for testing

Spot Quiz

Brighter Blue

02

A deviation from the specified or expected behavior that is visible to all instances of end-users as users is called:

A An error

B A fault

C A failure

D A defect

Spot Quiz

03

The later in the development life cycle a fault is discovered, the more expensive it is to fix. Why?

A The documentation is poor, so it takes longer to find out what the software is doing.

B Wages are rising.

C The fault has been built into more documentation, code, tests, and so on.

02 Regression Testing

Software regression testing

Software regression testing

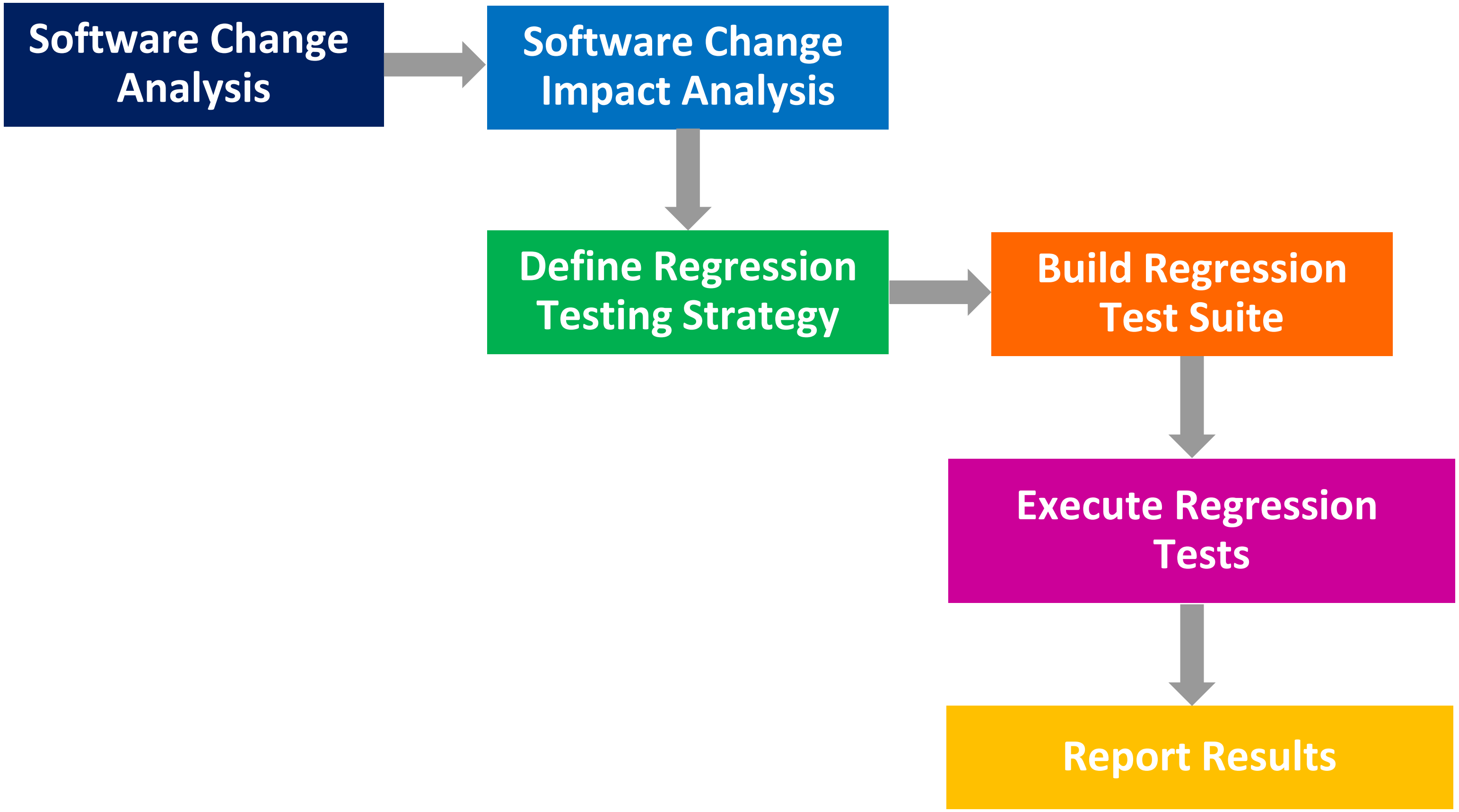
- It verifies that no unwanted changes were introduced to one part of the system as a result of making changes to another part of the system.
- It is done to make sure that new code changes do not have side effects on the existing functionalities.
- It ensures that the old code still works after the new code changes are done.
- It is nothing but full or partial selection of already executed test cases which are re-executed to ensure existing functionalities are working fine.

Regression Test Selection

The selection of test cases for regression testing depends on the following factors:

1. Scope of the bug fixes
2. Area of frequent defects
3. Area which has undergone many / recent code changes
4. Area which is highly visible to the users
5. Core features of the software which are mandatory requirements of the customers

Regression testing process



Software Change Analysis

Software Change Impact Analysis

Define Regression Testing Strategy

Build Regression Test Suite

Execute Regression Tests

Report Results

Problems and challenges in software regression testing

Major regression testing problems

- How to use a systematic method or tool to identify changed software parts
- How to use a systematic method or tool to identify software change impacts
- How to use a systematic method or tool to identify affected software test cases
- How to reduce the re-test suites
- How to select the test cases in a test suite

Major challenge in software regression testing

How to minimize re-testing efforts and achieve the adequate testing coverage?

Questions?

Spot Quiz

01

The difference between re-testing and regression testing is:

A

Re-testing ensures the original fault has been removed; **regression testing** looks for unexpected side-effects.

B

Re-testing looks for unexpected side-effects; **regression testing** ensures the original fault has been removed.

C

Re-testing is done after faults are fixed; **regression testing** is done earlier.

D

Re-testing is done by developers; **regression testing** is done by independent testers.

Spot Quiz

02

Regression testing mainly helps to:

A Check side-effects of fixes

B Ensure high level sanity

C Retest fixed defects

D Check the core gaps

Spot Quiz

03

Regression testing is mainly used to:

1 Ensure that the old code still works after the new code changes are done.

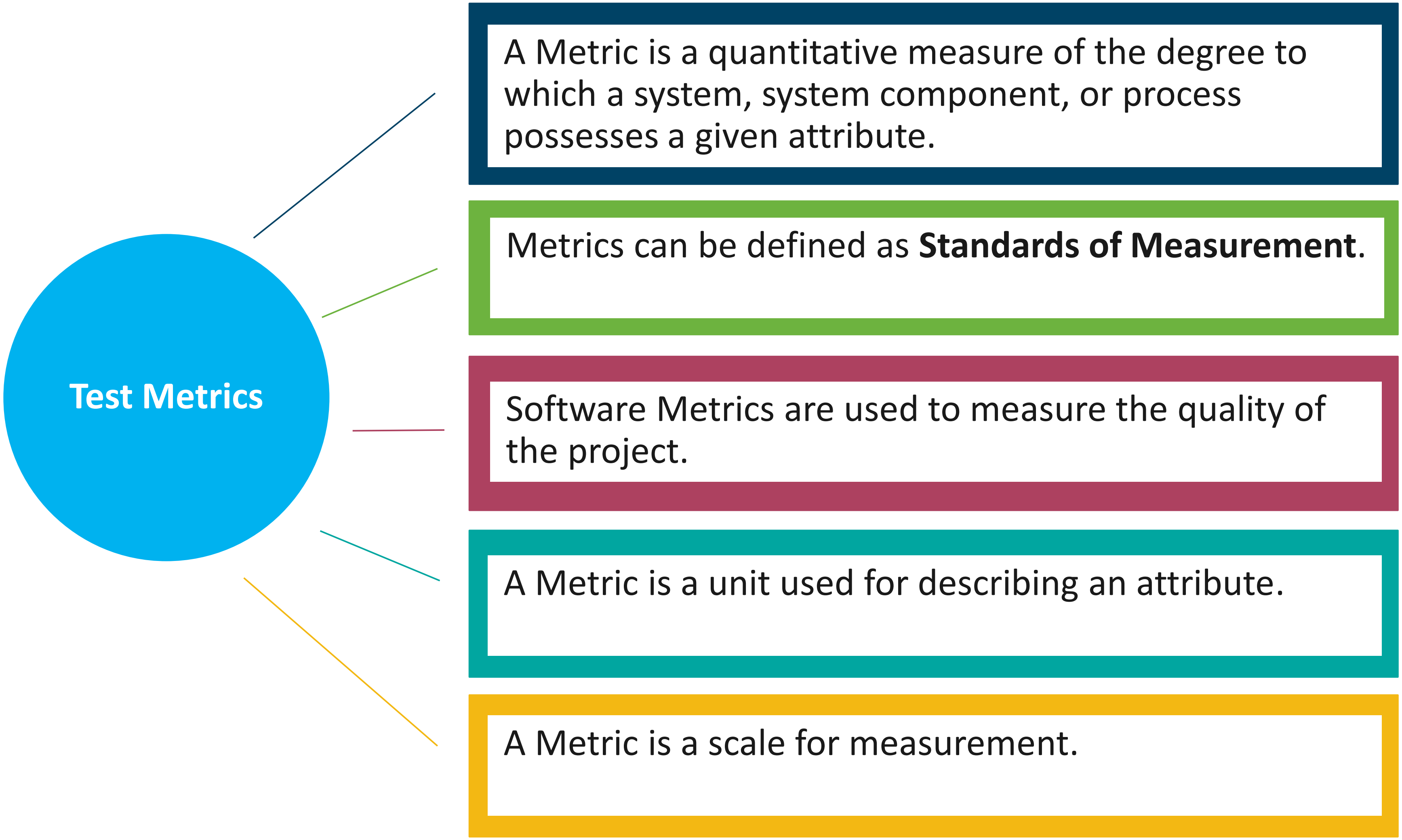
2 Do a simple check to see if the produced material is rational.

3 Understand the behavior of the system under a specific expected load.

4 Understand the upper limits of capacity within the system.

03 Test Metrics, Test Reports, and Sign-off

What is **Test Metrics**?



Test Metrics

A Metric is a quantitative measure of the degree to which a system, system component, or process possesses a given attribute.

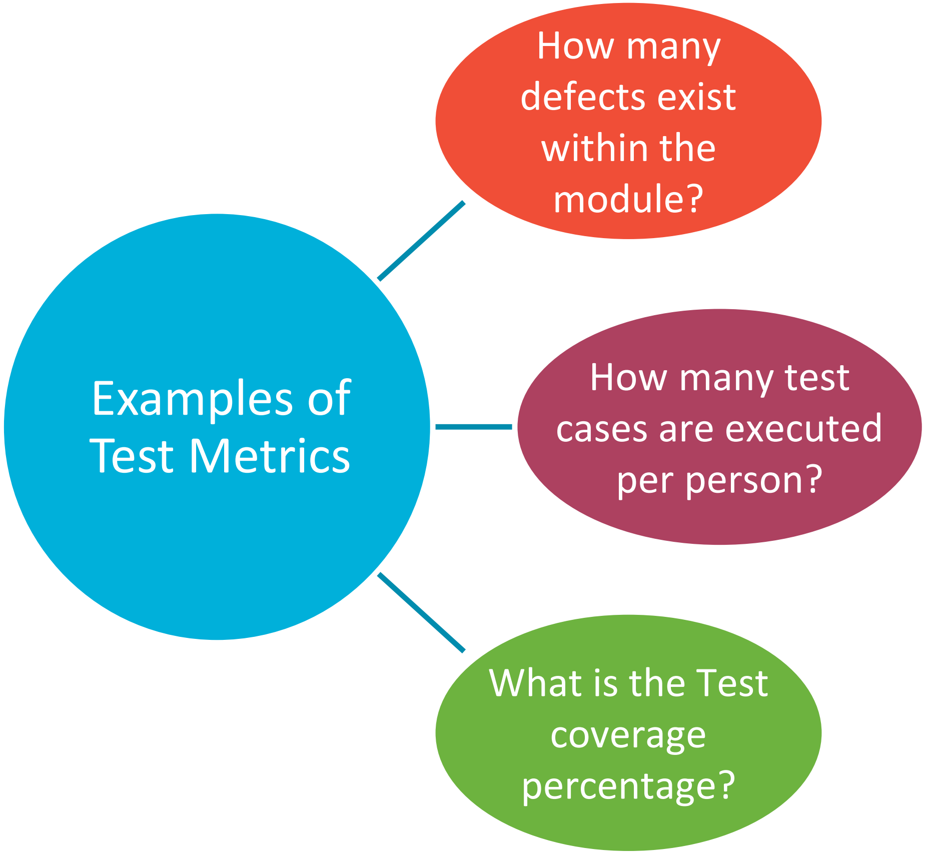
Metrics can be defined as **Standards of Measurement**.

Software Metrics are used to measure the quality of the project.

A Metric is a unit used for describing an attribute.

A Metric is a scale for measurement.

Examples of Test Metrics



Examples of Test Metrics

How many defects exist within the module?

How many test cases are executed per person?

What is the Test coverage percentage?

Measurement and Metrics

Measurement:

- A quantitative observation of an attribute or aspect of the software process, product, or project. In simple terms, it is nothing but a raw data. It constitutes:
  - Total number of test cases
  - Number of test cases executed
  - Number of test cases passed or failed, and so on
- **Example:** Number of lines of source codes, number of defects discovered during various tests, and so on.

Metrics:

- Metrics can be defined as a ratio of two measures, which are then used as meaningful indicators of processes or products. Simpler definition in metrics is a computed data. It constitutes:
  - Test case coverage
  - Test confidence
  - Test productivity and so on
- **Example:** Productivity = (Size / Effort)

Measurements and Metrics (continued)

Measures:

- Total number of test cases
- Number of test cases executed
- Number of test cases passed or failed, and so on

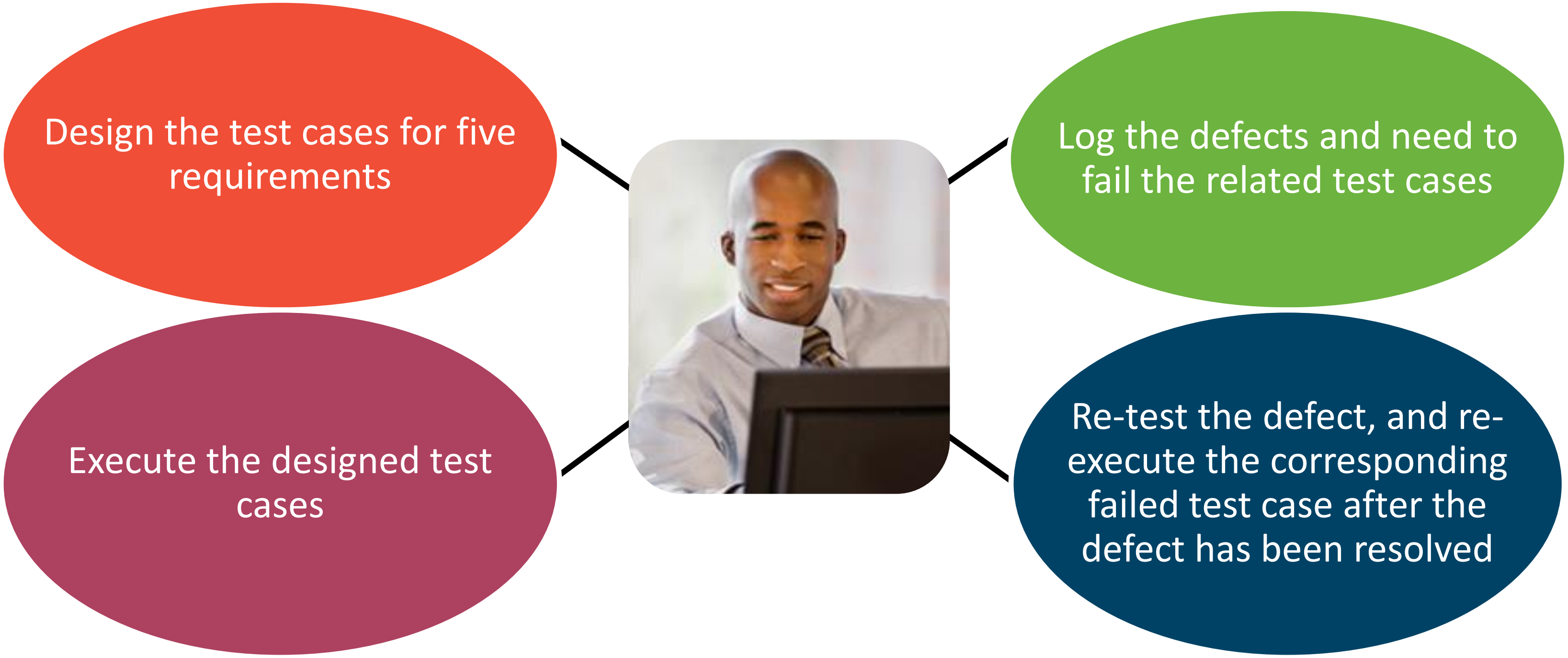
Metrics:

- Test case coverage
- Test confidence
- Test productivity and so on

Significance of Software Testing Metrics (1 of 3)

Test Metrics is important for measuring the quality of software. Suppose, a project does not have any metrics. Then, how is the quality of work done by a Test Analyst measured?

- A Test Analyst has to:



Design the test cases for five requirements

Log the defects and need to fail the related test cases

Execute the designed test cases

Re-test the defect, and re-execute the corresponding failed test case after the defect has been resolved

Significance of Software Testing Metrics (2 of 3)

If Metrics are involved in the project, then the exact status of work with proper numbers or data can be published.

In the test report, we can publish:

1. How many test cases have been designed per requirement?
2. How many test cases are yet to be designed?
3. How many test cases are executed?
4. How many test cases are passed / failed / blocked?
5. How many test cases are not yet executed?
6. How many defects are identified and what is the severity of those defects?
7. How many test cases have failed due to one particular defect?

Significance of Software Testing Metrics (3 of 3)

Based on the metrics (on the previous slide), test lead or manager will get the understanding of the below mentioned key points:

- Percentage (%) of work completed
- Percentage (%) of work yet to be completed
- Time to complete the remaining work
- Whether the project is going as per the schedule or lagging, and so on

Examples of Measurements

S. Number	Testing	Data retrieved during test case development and execution
1	Number of Requirements	5
2	Avg. Number of Test cases written per Requirement	20
3	Total Number of Test cases written for all Requirements	100
4	Total Number of Test cases Executed	65
5	Number of Test cases Passed	30
6	Number of Test cases Failed	26
7	Number of Test cases Blocked	9
8	Number of Test cases not executed	35
9	Total Number of Defects identified	30
10	Critical Defects count	6
11	High Defects Count	10
12	Medium Defects Count	6
13	Low Defects Count	8

Metrics Examples

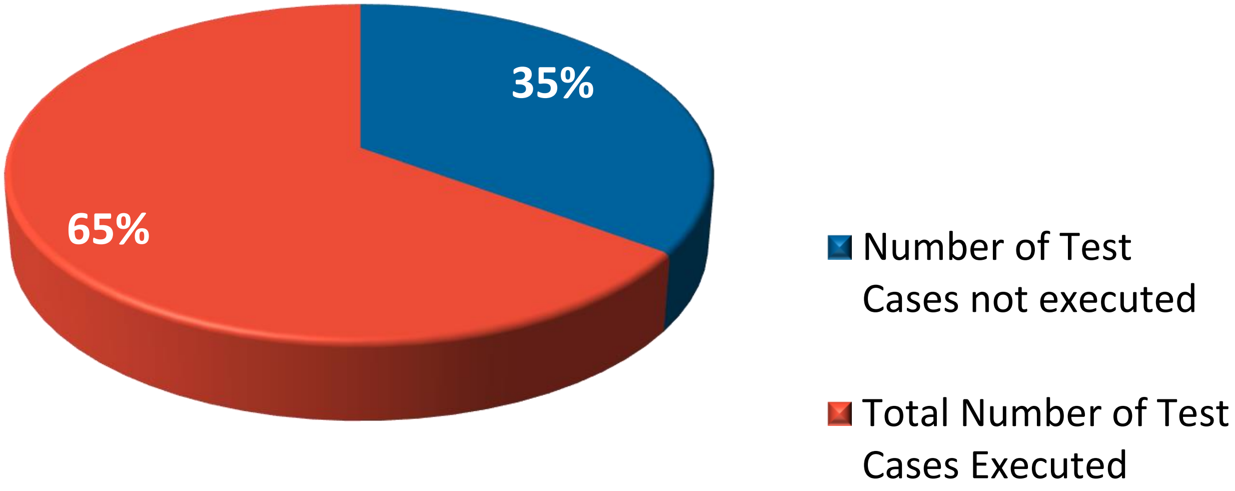
Percentage (%) Test Cases Executed: This metric is used to obtain the execution status of the test cases in terms of Percentage (%).

Percentage (%) Test cases Executed = (Number of Test cases executed / Total Number of Test cases written) * 100.

So, from the above data,
Percentage (%) Test cases Executed

= (65 / 100) * 100 = 65%

Test execution completion percentage (%)



Metrics Examples (continued)

Percentage (%) Test cases Passed / Failed / Blocked

Formula: Percentage (%) Test cases Passed / Failed / Blocked = (Number of Test cases Passed / Failed / Blocked / Total Number of Test cases Executed) * 100.

So, from the above data,
Percentage (%) Test cases Passed

= (30 / 65) * 100 = 46%

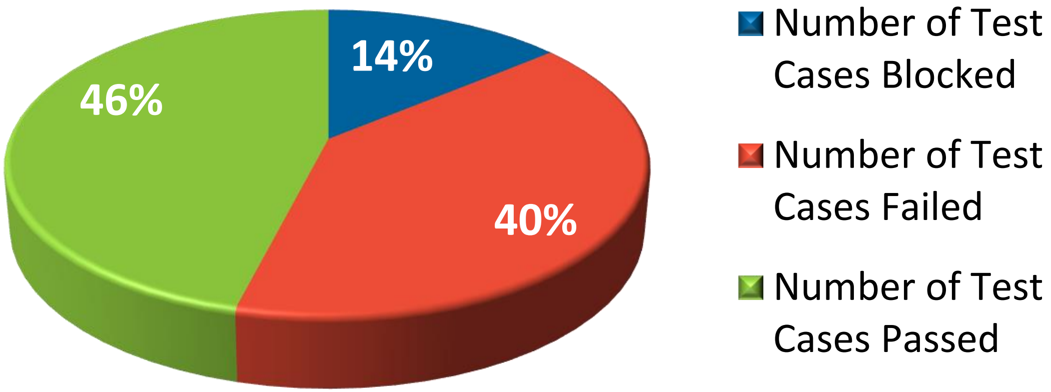
Percentage (%) Test cases Failed

= (26 / 65) * 100 = 40%

Percentage (%) Test cases Blocked

= (9 / 65) * 100 = 14%

Test execution status

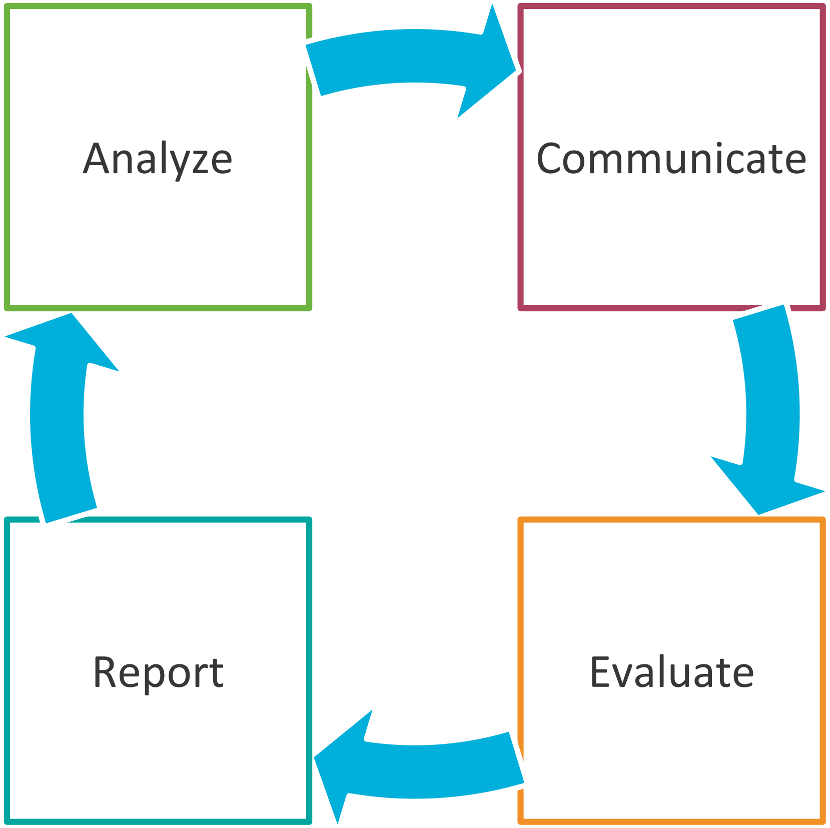


Benefits of Metrics

Metrics helps in:

- Controlling the project
- Monitoring the health of the project
- Ensuring project objectives and the IBM strategy are met
- Making data-driven decisions
- Identifying areas to drive process improvements
- Identifying point to do a root cause analysis
- Making prediction

Metrics Life Cycle



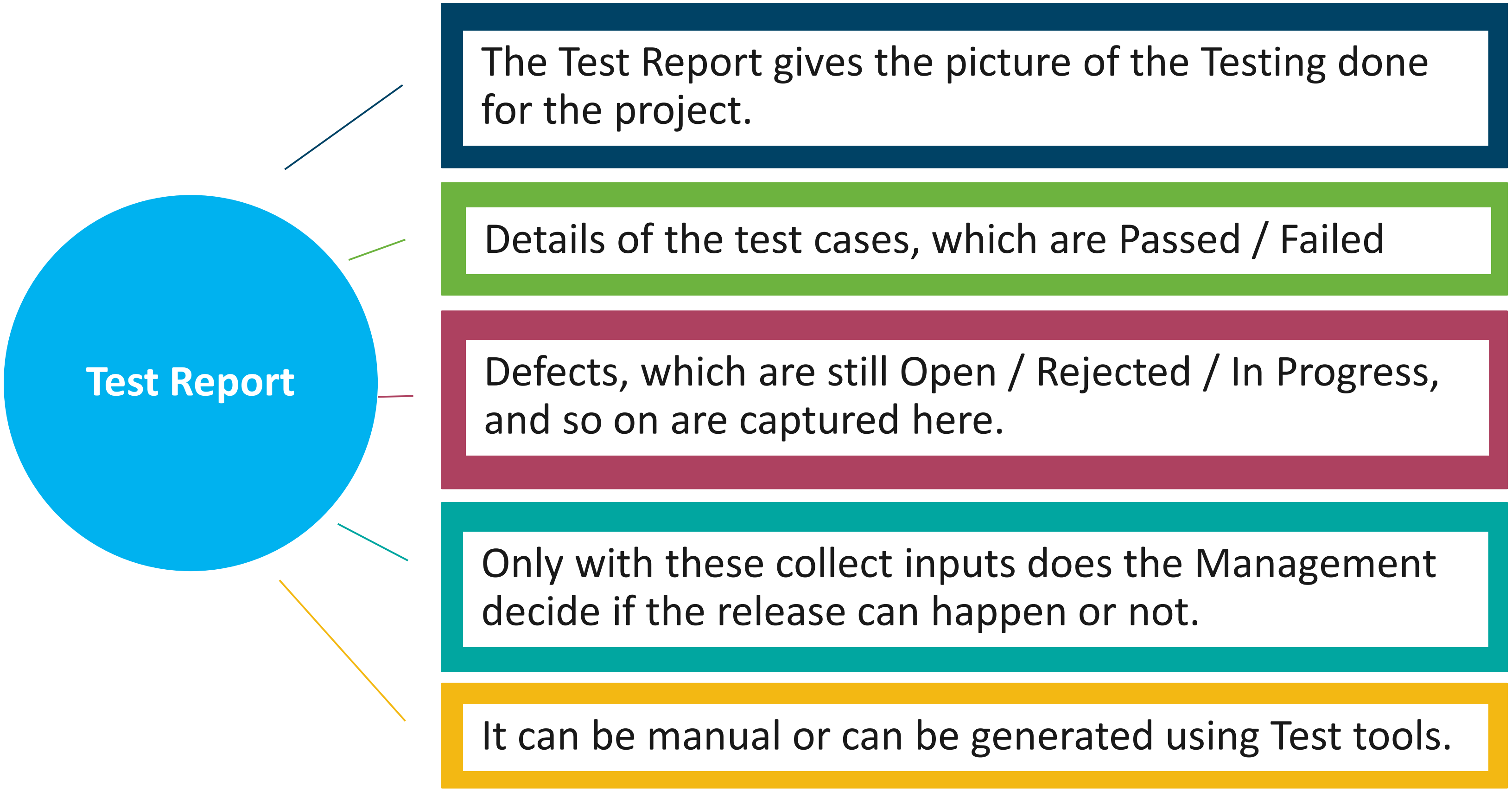
Analyze

Communicate

Evaluate

Report

Test Report



Test Report

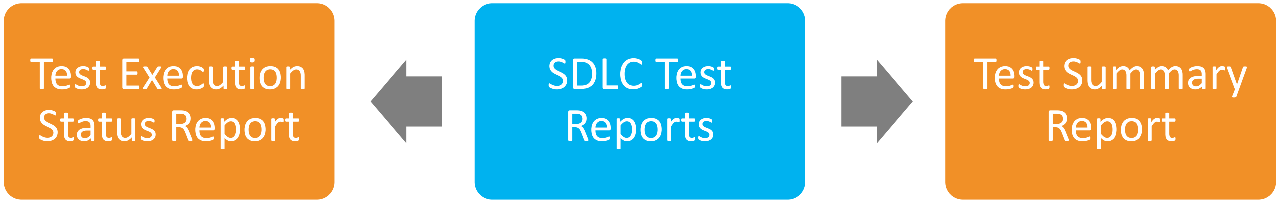
The Test Report gives the picture of the Testing done for the project.

Details of the test cases, which are Passed / Failed

Defects, which are still Open / Rejected / In Progress, and so on are captured here.

Only with these collect inputs does the Management decide if the release can happen or not.

It can be manual or can be generated using Test tools.



Test Execution Status Report

SDLC Test Reports

Test Summary Report

Test Execution Status Report (1 of 3)

Test Execution Status Report should contain the following **10 points**:

□ Number of test cases planned for that day

□ Number of test cases executed that day

□ Number of test cases executed overall

□ Number of defects encountered that day and their respective states

□ Number of defects encountered so far and their respective states

Test Execution Status Report (2 of 3)

- □ Number of critical defects that are still open
- □ Environment downtimes, if any
- □ Showstoppers, if any
- □ Attachment of the test execution sheet / Link to the test management tool where the test cases are placed
- □ Attachment to the bug report / link to the defect / test management tool used for incident management

You can view an example of a Test Execution Report on the next slide.

Test Execution Status Report (3 of 3)

Module	Scenarios	Sub Levels	Complexity	Responsible tester	Date of Execution (Can be past, present, of future date)	Status (Pass/Fail/Blocked/Not Executed)	Defect ID Brief description	Severity	Status
Admin	Login Page	Login	Medium	Tester A	31-08-2013	Pass			
Admin	Country Management	Add Country	Complex	Tester B	31-08-2013	Pass			
		Delete Country	Complex	Tester B	31-08-2013	Pass			
		Verify the list display	Medium	Tester B	31-08-2013	Pass			
Admin	City Management	Add City	Complex	Tester A	31-08-2013	Pass			
		Delete City	Complex	Tester A	31-08-2013	Pass			
		Verify the list display	Medium	Tester A	31-08-2013	Pass			
Admin	Car Make Management	Add Car Make	Complex	Tester B	31-08-2013	Fail	1028: Cannot add car make, 404 error on clicking the link	1- High	Open
		Delete car make	Complex	Tester B	31-08-2013	Blocked	1028		
		Verify the list display	Medium	Tester B	31-08-2013	Pass			

Test Summary Report (1 of 4)

Test Summary Report

- Test Summary Report is an important deliverable which is prepared at the end of a Testing project, or rather after Testing is completed.
- The prime objective of this document is to explain various details and activities about the Testing performed for the project to the respective stakeholders like senior management, client, and so on.
- As part of Test execution report, daily testing results are shared with involved stakeholders every day. But Test Summary Report provides a consolidated report on the Testing performed so far for the project.

Test Summary Report (2 of 4)

A Test Summary Status Report should contain the below sections:

Purpose of the document

Short description about the objective of preparing the document

Application Overview

Brief description about the application tested

Testing Scope

This section explains the functions / modules in scope and out of scope for testing; any item which is not tested due to any constraints / dependencies / restrictions.

Metrics

Metrics will help to understand the test execution results, status of test cases and defects, and so on.

Test Summary Report (3 of 4)

Types of testing performed

Describe the various types of Testing performed in a project.

Test Environment and tools

Provide details on Test Environment in which the Testing is carried out; Server, Database, Application URL, and so on.

Lessons Learned

This section is used to describe the critical issues faced and their solutions (how they were solved during Testing).

Recommendations

Any workaround or suggestions can be mentioned here.

Test Summary Report (4 of 4)

Best Practices

Activities regarded as best practices during the project can be documented as **Value Add** to showcase to the Stakeholders.

Exit Criteria

Exit Criteria is defined as a Completion of Testing by fulfilling certain conditions like all planned test cases are executed, all Critical defects are Closed, and so on.

Conclusion or Sign-Off

This section will mention whether the Testing team agrees and gives a Green signal for the application to **Go Live** or not, after the Exit Criteria was met.

Definitions, Acronyms, and Abbreviations

This section mentions the meanings of Abbreviated terms used in this document and any other new definitions.

Sign-off

Sign-Off

- This is a written statement from the test team to the client and to management saying that all the features are tested as per the requirement.
- In case there are few open issues / defects that still exist, and sign off is required, then conditional sign off is done with the details of the open issues / defects.

Sample Sign-off Email

To: Client, PM, Dev team, DB team, BA, QA team, Environment Team **(and anyone else that needs to be included)**

Hello Team,

The QA team signs-off on the Orange HRM version 3.0 software after the successful completion of the 2 cycles of functional testing the website.

The test cases and their execution results are attached to the email. **(Or mention the location where they are present. If using test management software, provide details regarding the same.)**

The list of known issues is attached to the email too. **(Again, any other references that make sense can be added.)**

Thanks,
QA team lead.

Activity

Create a Sign-Off email

Questions?

Spot Quiz

Brighter Blue

01

Which of the following is NOT a benefit of metrics?

A It helps in making data-driven decisions.

B It helps in monitoring the health of a project.

C It ensures how project objectives and the IBM strategy are met.

D It prohibits the testing team to control the project.

Spot Quiz

02

Which of the following is included in a test summary report?

A Names of the Testing Team

B Testing Scope

C Weekly Showstoppers

D Team Assignments

04 Case Study

Case Study: Day 9—Defect Management and Reporting

Let us get started with the some real life case studies now. Here is what you need to do:

- Work with your team as per instructions from the facilitator
- Discuss the various reporting options of RQM and defect reports to analyze the status of the test project among the team
- The observer will note down the key points from the discussion
- Share your key takeaways with the class based on the discussion (30 mins)

Microsoft Word
7 - 2003 Documen

Questions?