Defect Management

Errors

> An error is a mistake, misconception, or misunderstanding on the part of a software developer.

Faults (Defects)

> A fault (defect) is introduced into the software as the result of an error. It is an anomaly in the software that may cause it to behave incorrectly, and not according to its specification.

Failures

> A failure is the inability of a software system or component to perform its required functions within specified performance requirements [2].

What is Bug?

A bug is the consequence/outcome of a coding fault.

A **Defect in Software Testing** is a variation or deviation of the software application from end user's requirements or original business requirements. A software defect is an error in coding which causes incorrect or unexpected results from a software program which does not meet actual requirements. Testers might come across such defects while executing the test cases.

conditions occur the fault

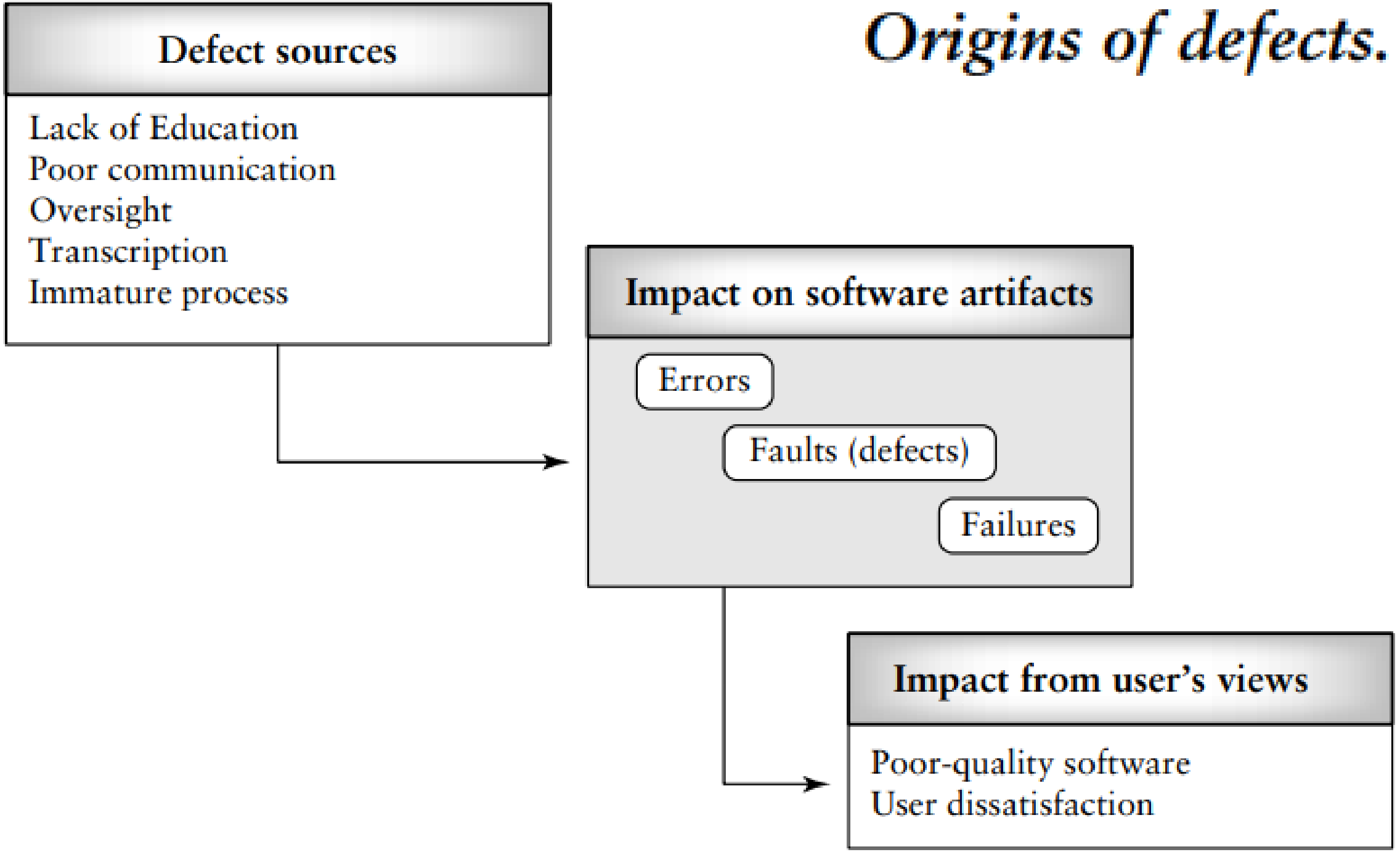
1. The input to the software must cause the faulty statement to be executed.
2. The faulty statement must produce a different result than the correct statement. This event produces an incorrect internal state for the software.
3. The incorrect internal state must propagate to the output, so that the result of the fault is observable.

A fault (defect) model can be described as a link between the error made (e.g., a missing requirement, a misunderstood design element, a typographical error), and the fault/defect in the software.

defect hypotheses

- design test cases;
- design test procedures;
- assemble test sets;
- select the testing levels (unit, integration, etc.) appropriate for the tests;
- evaluate the results of the tests.

Origins of defects.



Attributes	Description	Process
Activity	When did you detect the defect?	Open
Trigger	How did you detect the defect	
Impact	What would have customer noticed if defect had escaped into the field?	
Target	What high-level entity was fixed?	Close
Source	Who developed the target?	
Age	What is the history of the target?	
Defect type	What had to be fixed?	
Defect qualifier	Is the Defect Type missing, incorrect or extraneous?	

Bug Report

Attributes	Possible values
Defect_ID	The unique identification denoting the defect
Description	Description of the reported defect
Action_Generated	The action that generates the defect
Action_Removed	The action used to remove the defect
Severity	1: Exception, 2: Minor, 3: Average, 4: Major, 5: Critical
Defect_Type	0: None, 1: Calculation/logic, 2: Error checking, 3: Algorithm, 4: Func Interface, 6: Others
Target	The software development stage of the Action_Generated
Qualifier	0: Missing, 1: Incorrect, 2: Extraneous
Source	0: Produced, 1: Reused from library, 2: Outsourced

Defect Management Process

Defect Management is a systematic process to identify and fix bugs. A defect management cycle contains the following stages 1) Discovery of Defect, 2) Defect Categorization 3) Fixing of Defect by developers 4) Verification by Testers, 5) Defect Closure 6) Defect Reports at the end of project

Discovery

Discover defect

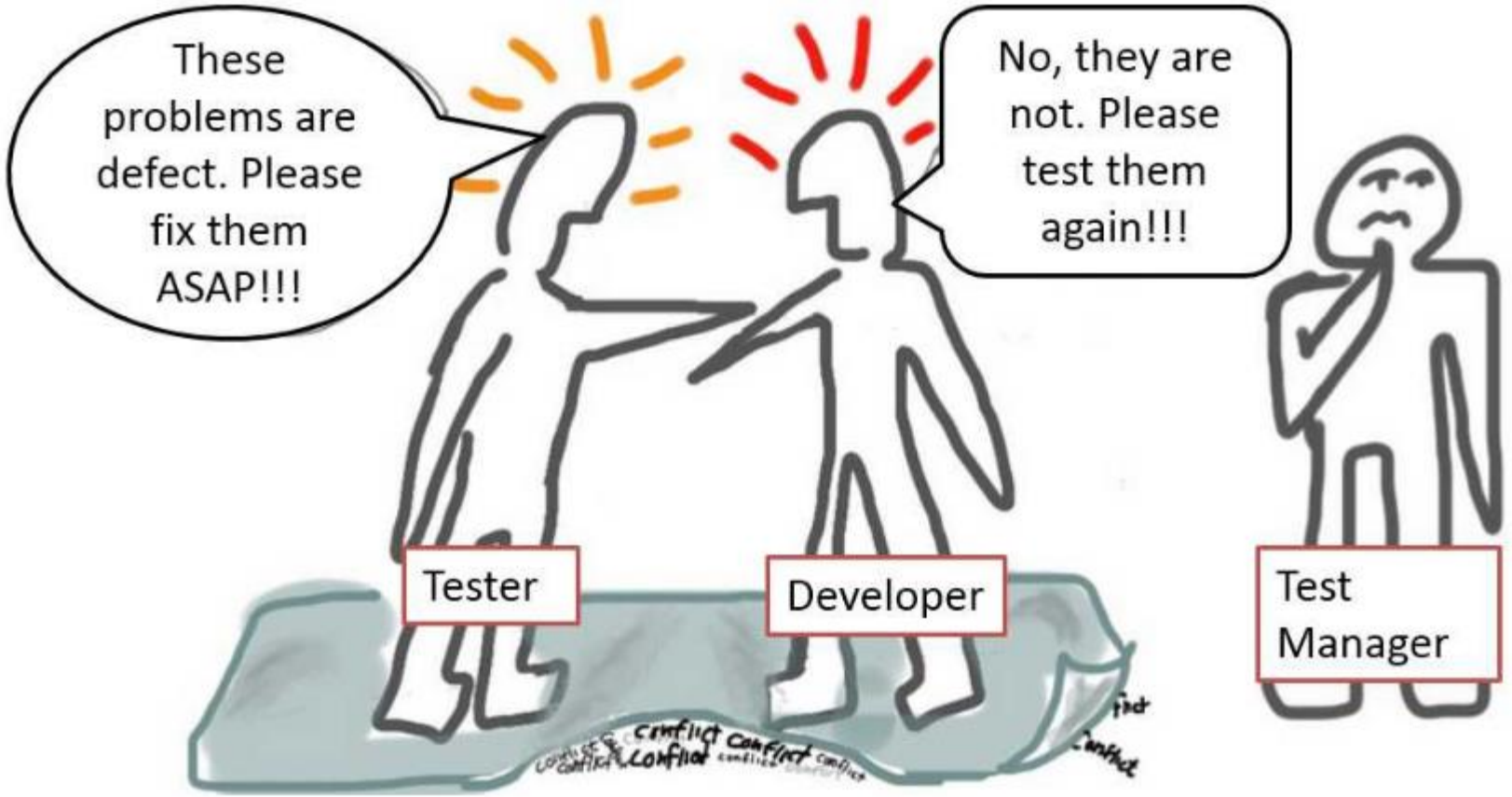
- Detect the defect before releasing to customer

Report defect

- Report the defect to development team

Accept defect

- Developer accepts or rejects the bug



Categorization

Critical

- The defects that need to be fixed **immediately** because it may cause great damage to the product

High

- The defect impacts the product's **main** features

Medium

- The defect causes **minimal** deviation from product requirement

Low

- The defect has **very minor** affect product operation

Defect Resolution

Defect Rejection Ratio

- (No. of defects rejected / Total no. of defects raised) * 100

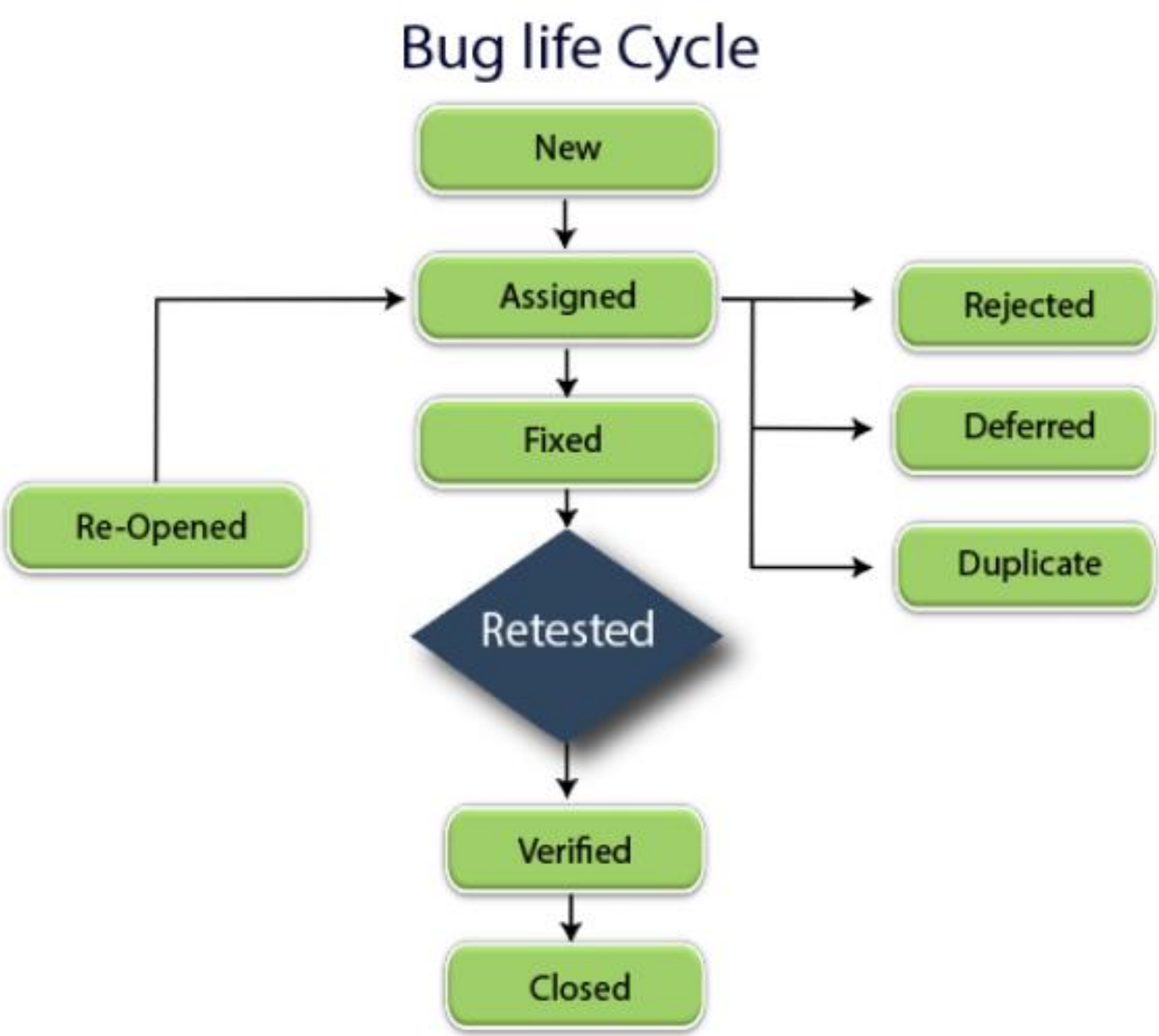
Defect Leakage Ratio

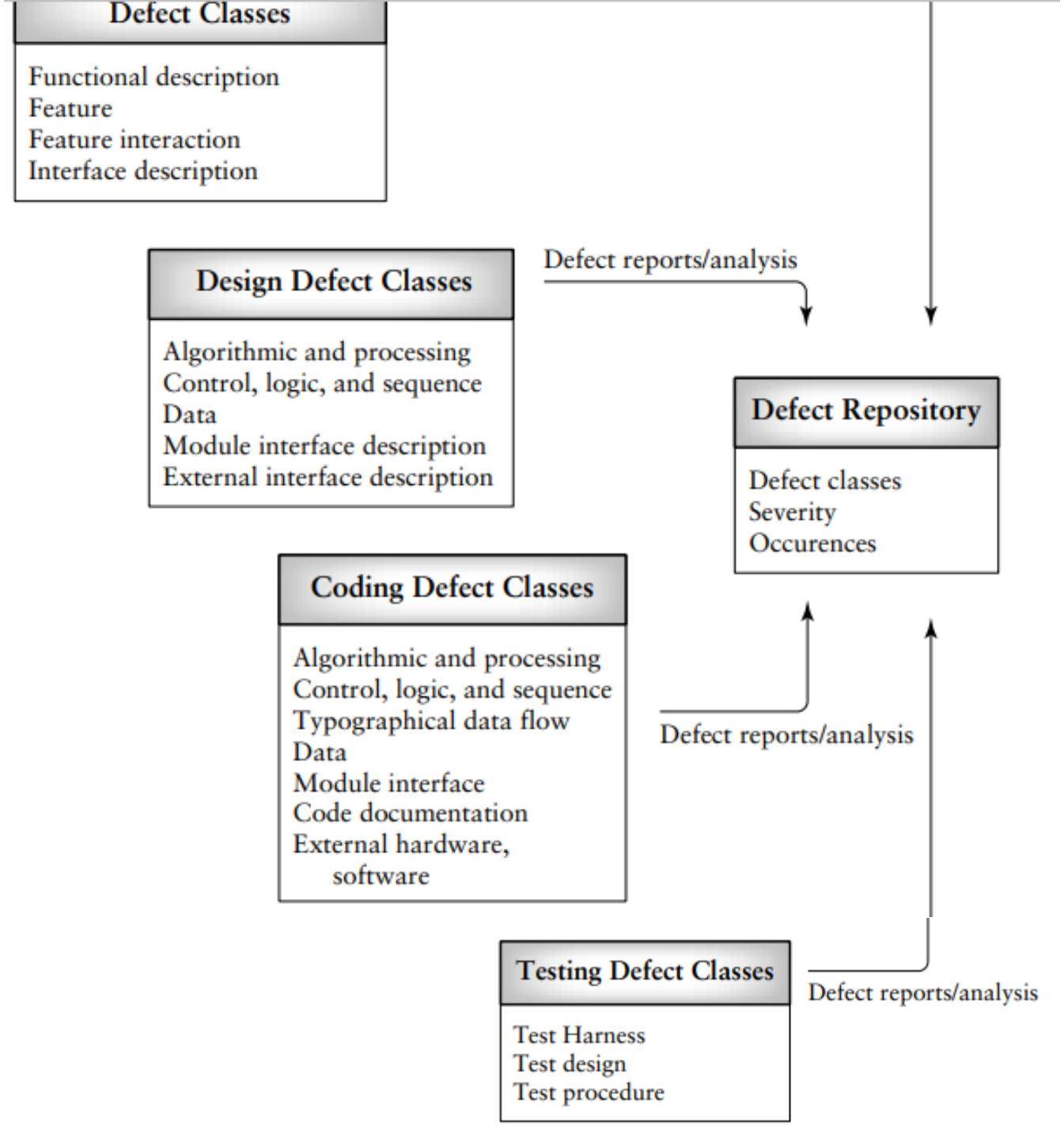
- (Number defect missed/ total defects of software) * 100

Suppose the XYZ Bank website has total **64** defects, but your testing team only detect **44** defects i.e. they missed **20** defects. Therefore, you can calculate the defect leakage ratio (DLR) is 20/64 = **0.312** (31.2 %).

For instance, end users found 42 faults at the UAT manufacturing unit, while 140 defects were found during the whole quality process, comprising both the testing facility and the production site. Defect leakage would thus be as follows:

= (42/140) * 100 = 30% 30% is the defect leakage percentage.


Bug life Cycle
New
Assigned
Rejected
Fixed
Deferred
Re-Opened
Duplicate
Retested
Verified
Closed



Defect classes and the defect repository.

calculates the total monetary value of a set of coins.

A sample specification with defects.

Specification for Program `calculate_coin_value`

This program calculates the total dollars and cents value for a set of coins. The user inputs the amount of pennies, nickels, dimes, quarters, half-dollars, and dollar coins held. There are six different denominations of coins. The program outputs the total dollar and cent values of the coins to the user.

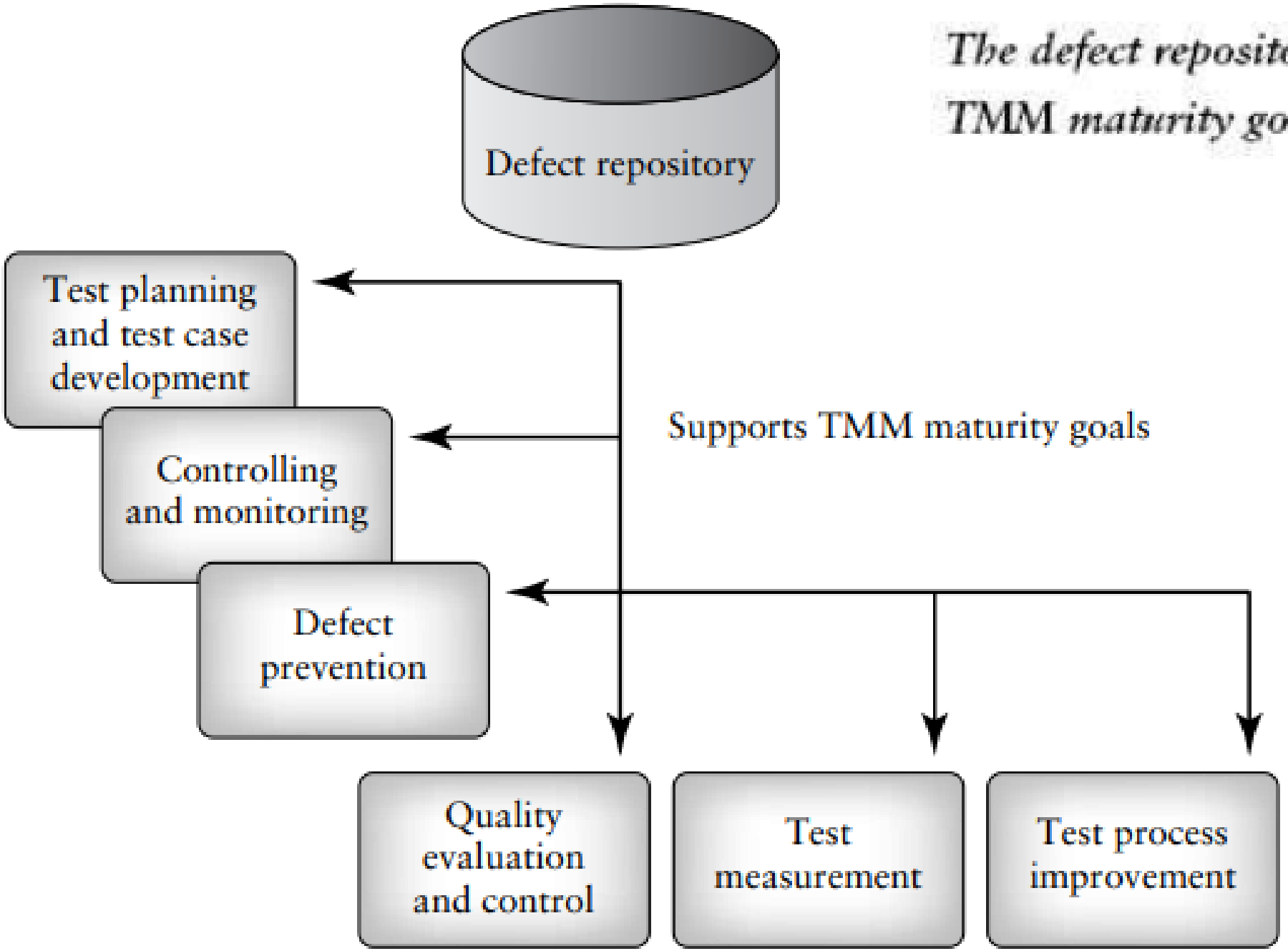
Inputs: `number_of_coins` is an integer
Outputs: `number_of_dollars` is an integer
`number_of_cents` is an integer

Data defects. This defect relates to an incorrect value for one of the elements of the integer array, coin_values, which should read 1,5,10,25,50,100.

A code example with defects.

```
/*****************************************************************
program calculate_coin_values  calculates the dollar and cents
value of a set of coins of different dominations input by the user
denominations are pennies, nickels, dimes, quarters, half dollars,
and dollars
*****************************************************************/
main ()
{
int total_coin_value;
int number_of_coins = 0;
int number_of_dollars = 0;
int number_of-cents = 0;
int coin_values = {1,5,10,25,25,100};
{
int  i = 1;
while ( i < 6)
{
      printf("input number of coins\n");
      scanf ("%d", number_of_coins);
      total_coin_value = total_coin_value +
       (number_of_coins * coin_value{i]);
}
i = i + 1;
number_of_dollars =  total_coin_value/100;
number_of_cents = total_coin_value - (100 * number_of_dollars);
printf("%d\n", number_of_dollars);
printf("%d\n", number_of-cents);
}
```

The defect repository, and support for TMM maturity goals.



calculates the total monetary value of a set of coins.

> **A precondition is a condition that must be true in order for a software component to operate properly.**

In this case a useful precondition would be one that states for example:

$$number\_of\_coins >= 0$$

> **A postcondition is a condition that must be true when a software component completes its operation properly.**

A useful postcondition would be:

$$number\_of\_dollars,\ number\_of\_cents >= 0.$$

Defect Tracking

It provides a way to remember all the defects found.

Allows the tester to concisely impart information regarding how they were found and what the issue is.

Improves efficiency by helping testers prioritize defects based on time, urgency, and various other factors of importance.

A defect tracker can be helpful in identifying the cause behind defects as well since log files and reports or descriptions are generally included to elaborate on the defect.

The tracker can also be used to log any updates or changes made to address the defect. This allows testers to keep track of corrections and retest to check whether the defect was fixed or update any additional defects found.

A tracker also helps the software testing team to gather metrics regarding defects and create preventative measures for avoiding such issues in the future.

Lastly, the defect tracker creates a well oiled, efficient testing workflow that saves a lot of time.

Bugzilla

Jira

Plan, track, and release world-class software

userback

ClickUp

axosoft

- Jira
- Bugzilla
- BugNet
- Redmine
- Mantis
- Trac
- Backlog

Selenium (Web Application Testing)

Appium (Mobile Testing)

JMeter (Load Testing)

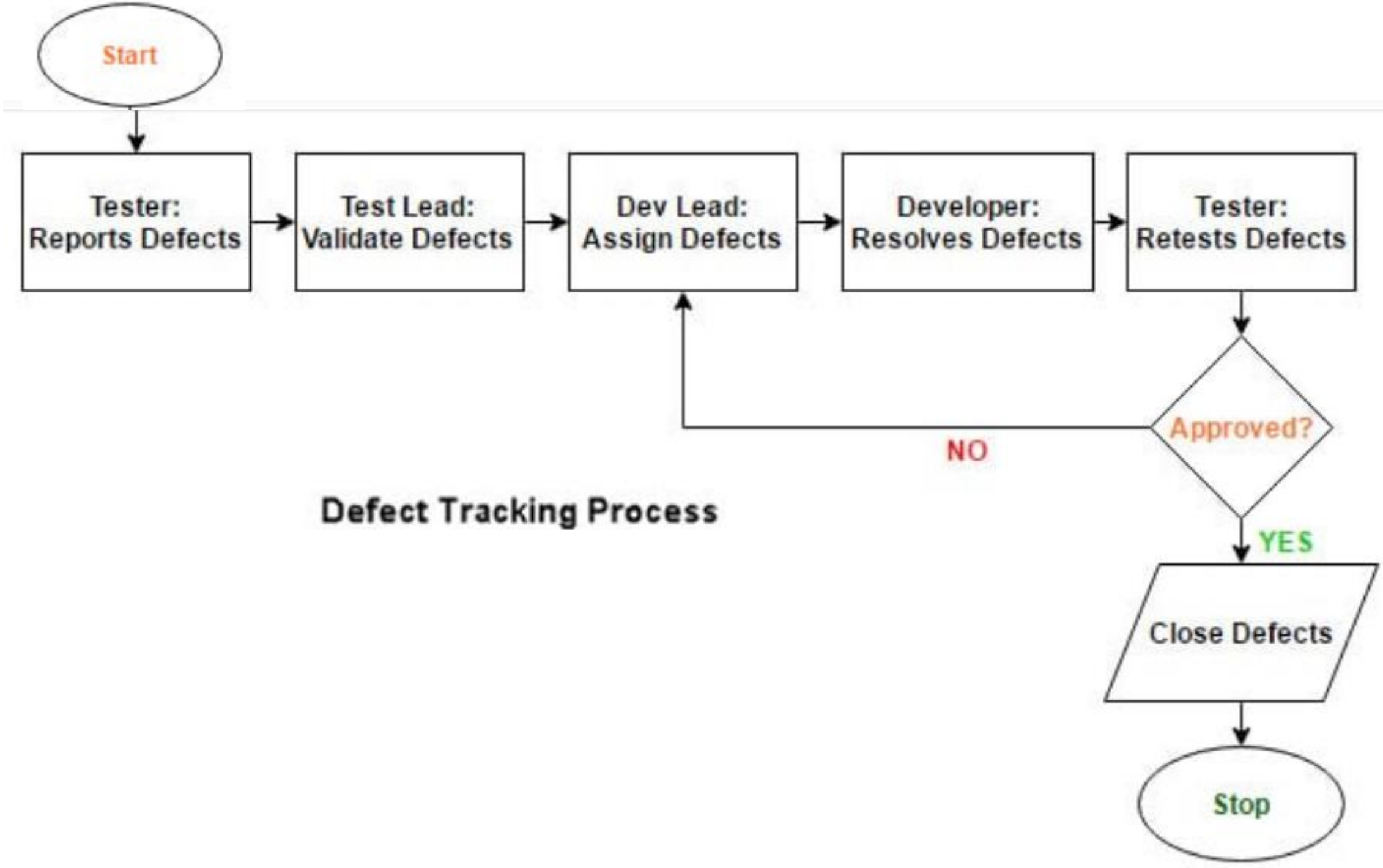
Jenkins (Continuous Testing)

TestLink (Test Management)

Mantis (Bug-Tracking & Project Management)

Postman (API Testing)

Firebug / Firepath (Online Debugging)



Defect Tracking Process

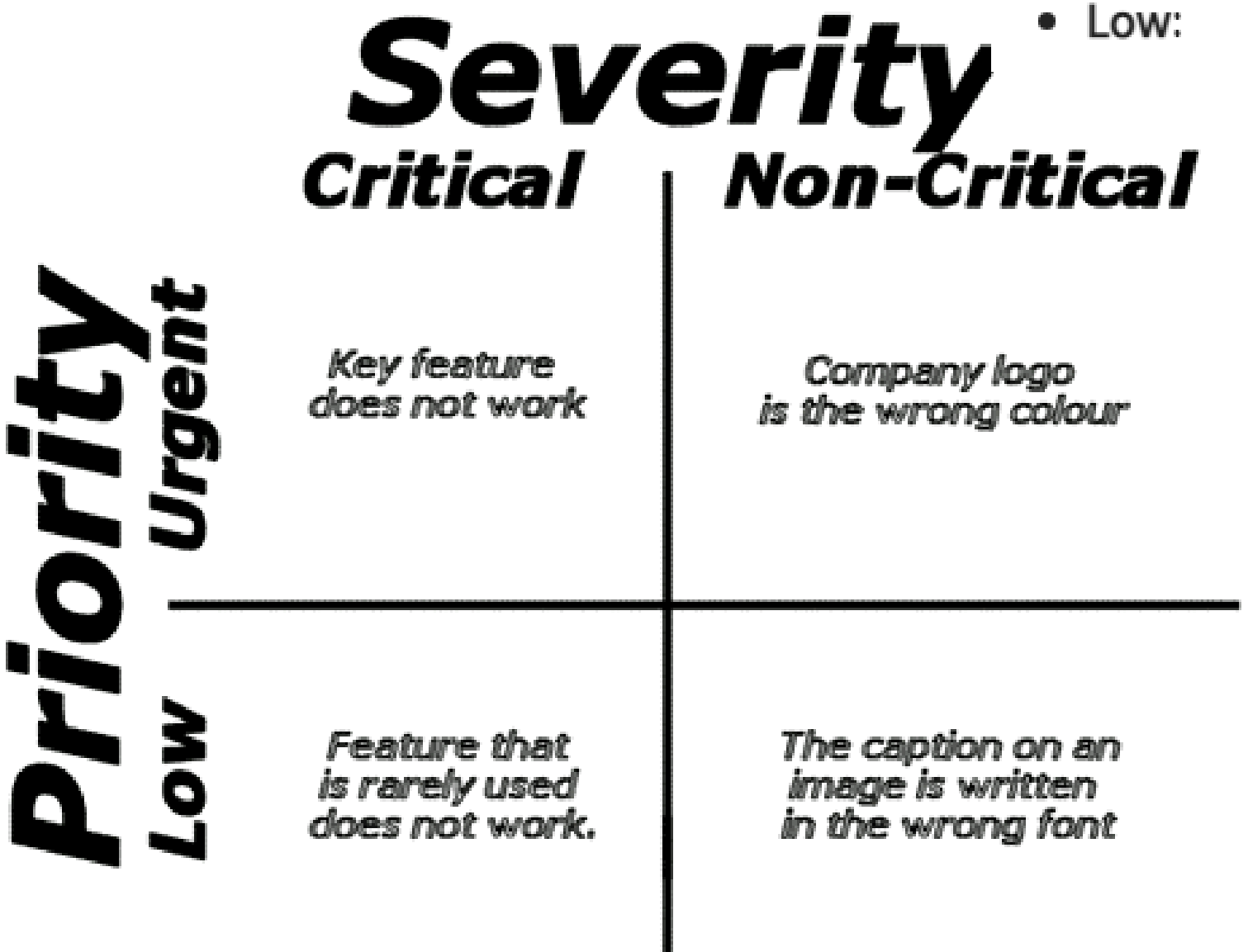
Severity and Priority in Testing

- Critical:
- Major:
- Medium:
- Low:

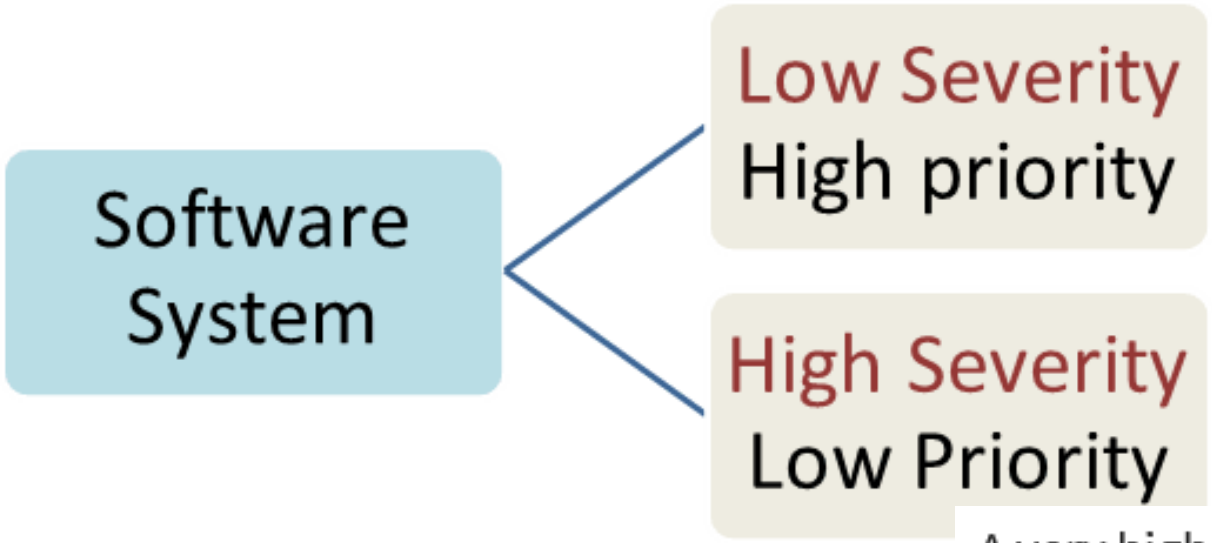
Severity

- Low:
- Medium:
- High:

Priority



Example of Defect Severity and Priority

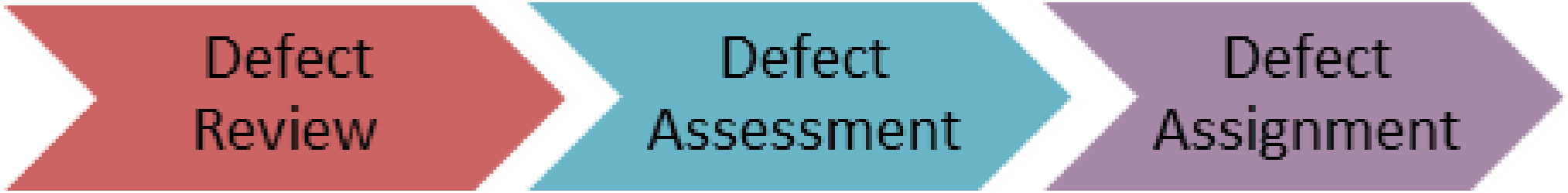


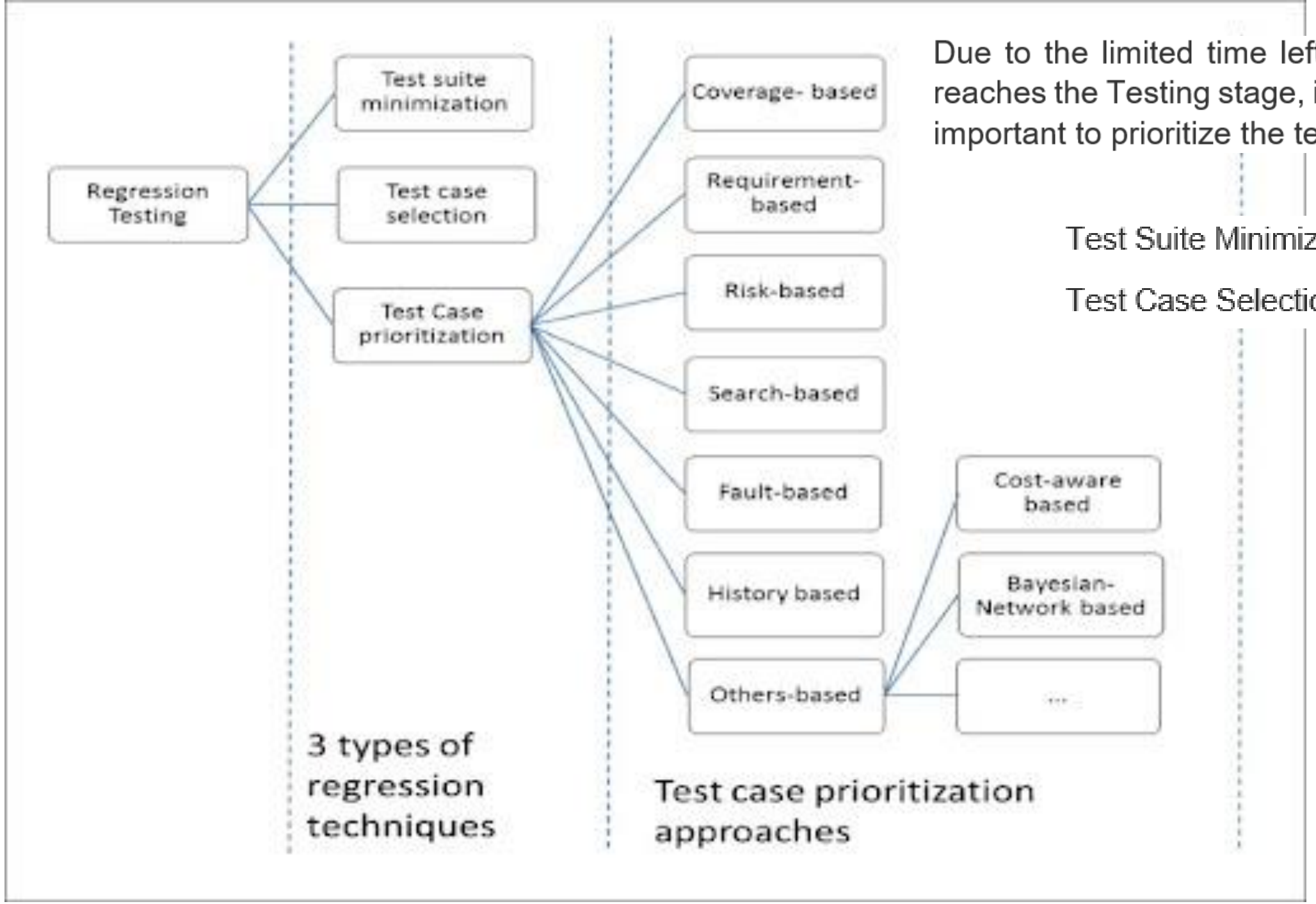
A logo error for any shipment website, can be of low severity as it not going to affect the functionality of the website but can be of high priority as you don't want any further shipment to proceed with the wrong logo

A very high severity with a low priority: Likewise, for **flight** operating website, a defect in reservation functionality may be of high severity but can be a low priority as it can be scheduled to release in a next cycle.

Defect Triage

Defect triage is a process that tries to do the re-balancing of the process where the test team faces the problem of limited availability of resources It provides a way to remember all the defects found.





Due to the limited time left when the product reaches the Testing stage, it has become more important to prioritize the test cases

Test Suite Minimization (TSM) and

Test Case Selection (TCS).

- **Priority 1:** The test cases, which MUST be executed, else the consequences may be worse after the product is released. These are critical test cases where the chances of getting a functionality disrupted due to a new feature is most likely possible.
- **Priority 2:** The test cases COULD be executed if there is enought time. These are not very critical cases, but can be executed as a best practice for a double check before launch.
- **Priority 3:** The test cases are NOT important to be tested prior to the current release. These can be tested later, shortly after the release of the current software version as a best practice. However, there is no direct dependency on it.
- **Priority 4:** The test cases are never important, as their impact is nearly negligible.

(Average Percentage of Faults Detected) that can be calculated using the below formula

$$APFD = 1 - ((TF1 + TF2 + ... + TFm) / nm) + 1 / 2n$$

where,

TFi = position of the first Test case in Test suite T that exposes Fault i

m = total number of Faults exposed under T

n = total number of Test cases in T

AFPD values can range from 0 to 100. Higher the APFD value, faster the fault detection rate

2. Programmer A and Programmer B are working on a group of interfacing modules. Programmer A tends to be a poor communicator and does not get along well with Programmer B. Due to this situation, what types of defects are likely to surface in these interfacing modules? What are the likely defect origins?

3. Suppose you are a member of a team that was designing a defect repository. What organizational approach would you suggest and why? What information do you think should be associated with each defect? Why is this information useful, and who would use it?

6. Suppose you are testing a code component and you discover a defect: it calculates an output variable incorrectly. (a) How would you classify this defect? (b) What are the likely causes of this defect? (c) What steps could have been taken to prevent this type of defect from propagating to the code?

7. Suppose you are testing a code component and you find a mismatch in the order of parameters for two of the code procedures. Address the same three items that appear in question 6 for this scenario.