What is Software testing?

Software testing is a part of software development lifecycle, it's aim is to ensure that the code to be deployed is of high quality with no bugs and no logical errors

1

Software Testing Classification

Testing type:
Manual
Automatic

Testing methods:
Static
Dynamic

Testing approaches:
Black Box
White Box
Gray Box

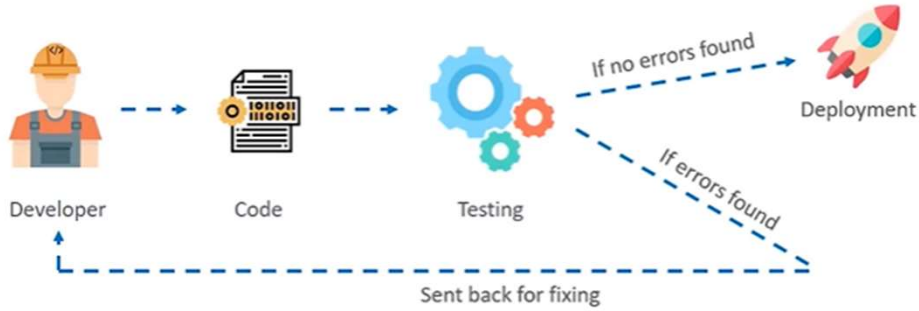
2

Testing levels:
Unit Testing
Integration Testing
System Testing
Acceptance Testing

3

Dynamic Testing

Dynamic testing Happens during the execution of the code. It can help identify subtle defects or vulnerabilities because it also looks at the code's integration with other databases, servers, and services.



4

Example Of Dynamic Testing Tools

Selenium

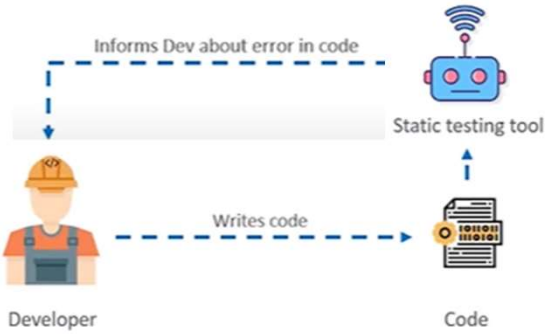
Katalon

CasperJS

5

Static Testing

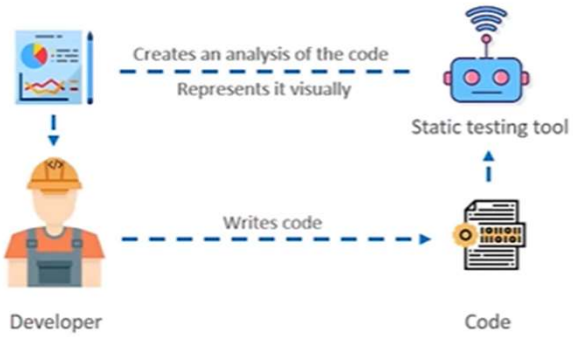
It is a method of debugging by examining source code before program is run, i.e. test the code without actually executing it. It does so by analyzing the code against a pre-set of coding rules and ensure that it conforms to the guidelines.



6

Static Code Analysis

There are many tools which help in static testing and providing us with a analysis for better comprehension.



7

Reasons To Use Static Code Analysis

- Finds errors earlier in development
- Detects overcomplexity in code (refactoring/simplification)
- Finds Security Errors
- Enforces Best Coding Practices
- Automated & Integrates in Jenkins
- Can create project specific rules

8

Example Of Static Testing Tools

SonarQube Coverity

9

What is SonarQube?

10

It's an open source static testing analysis software. It is used by developers to manage source code quality and consistency. Some of the code quality checks are:

- Potential bugs
- Code defects to design inefficiencies
- Code duplication
- Lack of test coverage
- Excess complexity

11

Features of SonarQube

It can work with 25 different languages.

12

Tricky Issues

- Detect Bugs
- Code Smells
- Security Vulnerability
- Activate Rules Needed
- Execution Path

SonarQube can detect tricky bugs or can raise issues on pieces of code that it thinks is faulty.

13

Tricky Issues

- Detect Bugs
- Code Smells
- Security Vulnerability
- Activate Rules Needed
- Execution Path

Code smells are the characteristics of a code that indicates that there might be a problem caused by the code in the future.

But smells aren't necessarily bad, sometimes they are how a functionality works and there is nothing that can be done about it

14

Tricky Issues

- Detect Bugs
- Code Smells
- Security Vulnerability
- Activate Rules Needed
- Execution Path

SonarQube can detect security issues that a code may face.

E.g. If a developer forgets to close an open a SQL database OR If important details like username and password have been directly written in the code.

15

Tricky Issues

- Detect Bugs
- Code Smells
- Security Vulnerability
- Activate Rules Needed
- Execution Path

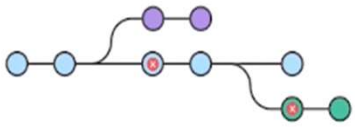
You can create and maintain different sets of rules that are specific to particular projects, these are known as Quality Profiles

16

Tricky Issues

- Detect Bugs
- Code Smells
- Security Vulnerability
- Activate Rules Needed
- Execution Path

Whenever there is Data flow in your program, and there is a lot of involvement between the different Modules. SonarQube can figure out if there are any tricky bugs in these execution paths.



17

Features of SonarQube

Enhanced Workflow(Ensure Better CI/CD)

- Automated Code Analysis
- Get access through Webhooks & API
- Integrate GitHub
- Analyze branches and Decorate pull requests

Keeps working in the background from the development phase itself, monitoring and identifying errors.

18

Enhanced Workflow

- Automated Code Analysis
- Get access through Webhooks & API
- Integrate GitHub
- Analyze branches and Decorate pull requests

To initiate tests we do not need to come to SonarQube directly, we can do that through an API call.

19

Enhanced Workflow

- Automated Code Analysis
- Get access through Webhooks & API
- Integrate GitHub
- Analyze branches and Decorate pull requests

It can be directly integrated with your choice of version control software.
E.g. GitHub

20

Enhanced Workflow

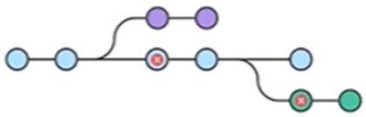
Automated Code Analysis

Get access through Webhooks & API

Integrate GitHub

Analyze branches and Decorate pull requests

It gives us a branch Level analysis, i.e. it doesn't just analyze the master branch it also analyzes the other branches, identifying any errors.



21

Features of SonarQube

Built-in methodology

Discover memory leaks

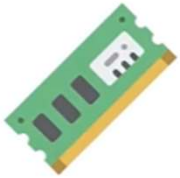
Good Visualizer

Enforces a quality gate

Digs into issues

Plugins for IDEs

It can show the memory leak in your application, if the application has a tendency to fail or go out of memory. This generally will happen slowly happen over a period of time



22

Built-in methodology

- Discover memory leaks
- Good Visualizer
- Enforces a quality gate
- Digs into issues
- Plugins for IDEs

it has a good way of visualizing, it gives simple overviews of the overall health of the code.

23

Built-in methodology

- Discover memory leaks
- Good Visualizer
- Enforces a quality gate
- Digs into issues
- Plugins for IDEs

It can enforce a quality gate, you can tell SonarQube based on your requirements and practices what code is wrong and what is correct.

24

Built-in methodology

- Discover memory leaks
- Good Visualizer
- Enforces a quality gate
- Digs into issues
- Plugins for IDEs

if it shows that there is a problem SonarQube allows you to go and directly check it out from the summary report or from one code file to another.

25

Built-in methodology

- Discover memory leaks
- Good Visualizer
- Enforces a quality gate
- Digs into issues
- Plugins for IDEs

It has plugin called 'SonarLint' which helps SonarQube to integrate itself with an IDE. Which means there is no need to install the whole SonarQube package.

26