
sonarqube

What is Software testing?

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

Software Testing Classification

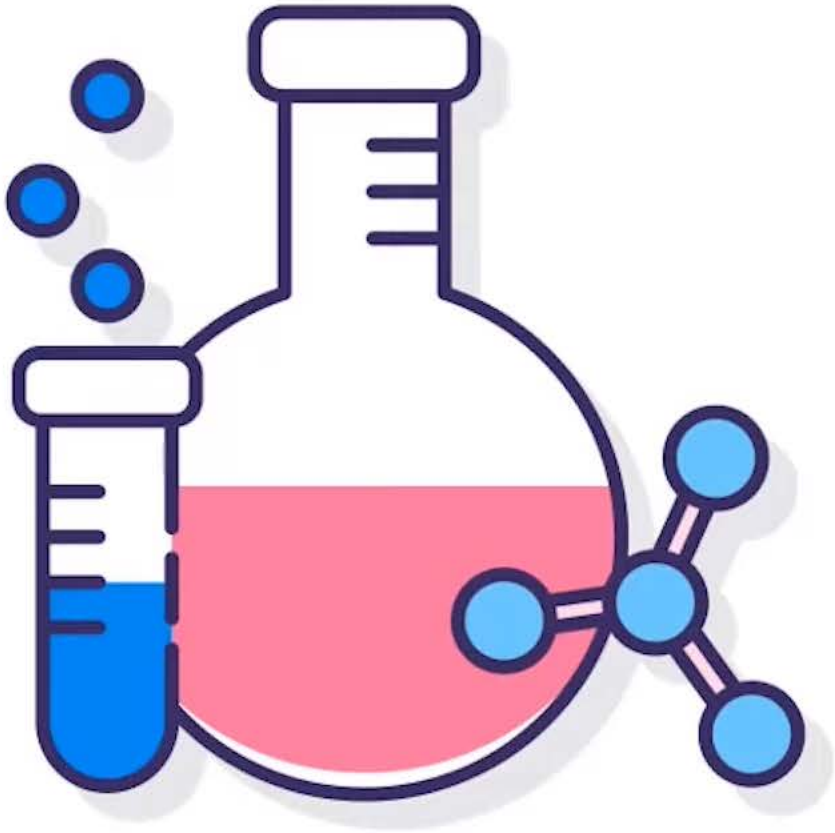
Testing type:
Manual
Automatic

Testing methods:
Static
Dynamic

Testing approaches:
Black Box
White Box
Gray Box

Software Testing Classification

Testing levels:
Unit Testing
Integration Testing
System Testing
Acceptance Testing



Dynamic Testing

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.

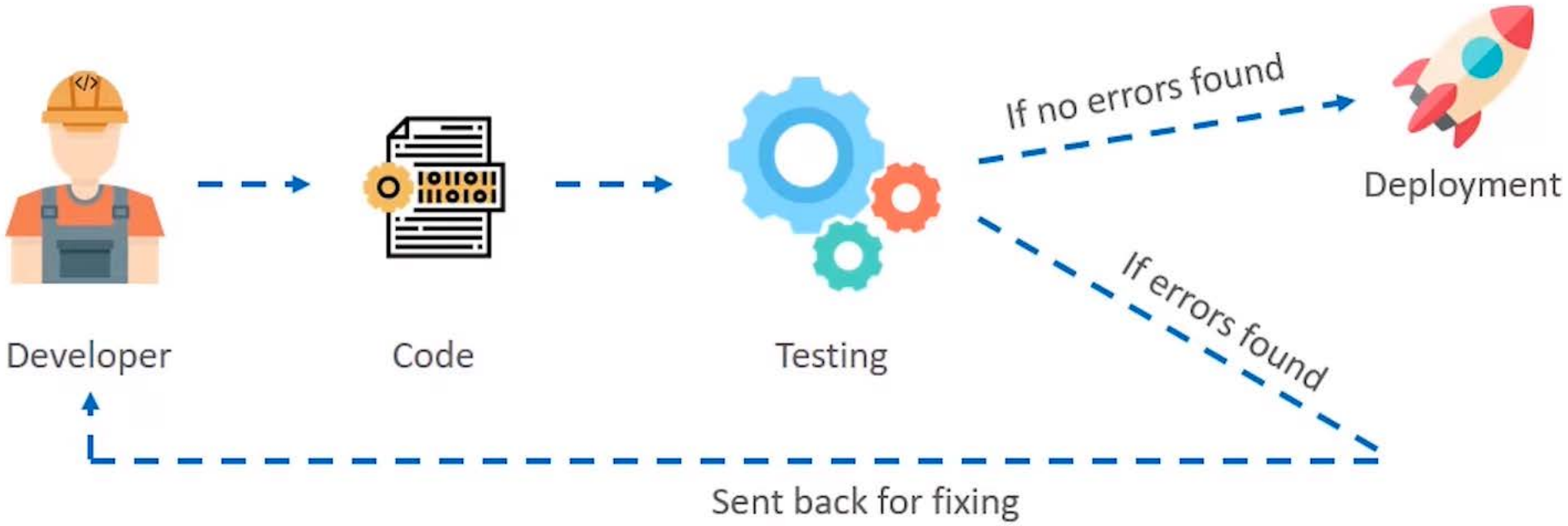
Developer

Code

Testing

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.

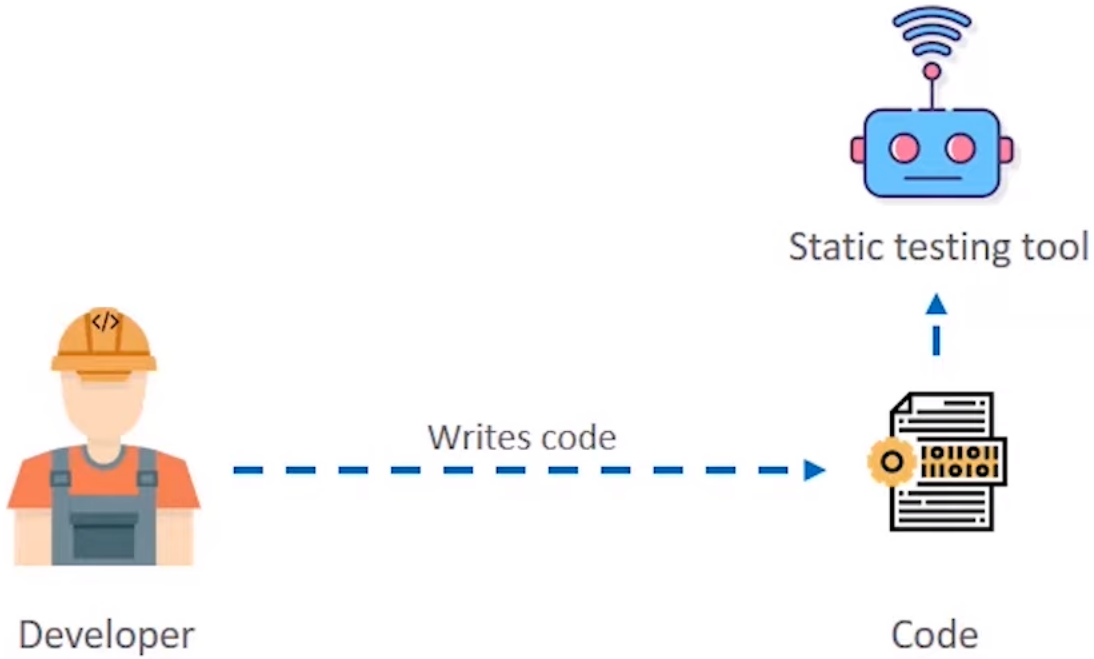


Advantages of Dynamic Testing

- It will find faults in the specific part of the code during the execution time.
- Some of the errors that wouldn't be found using static testing would definitely be found out using dynamic testing, especially those related to parts of the source code that rely on external services

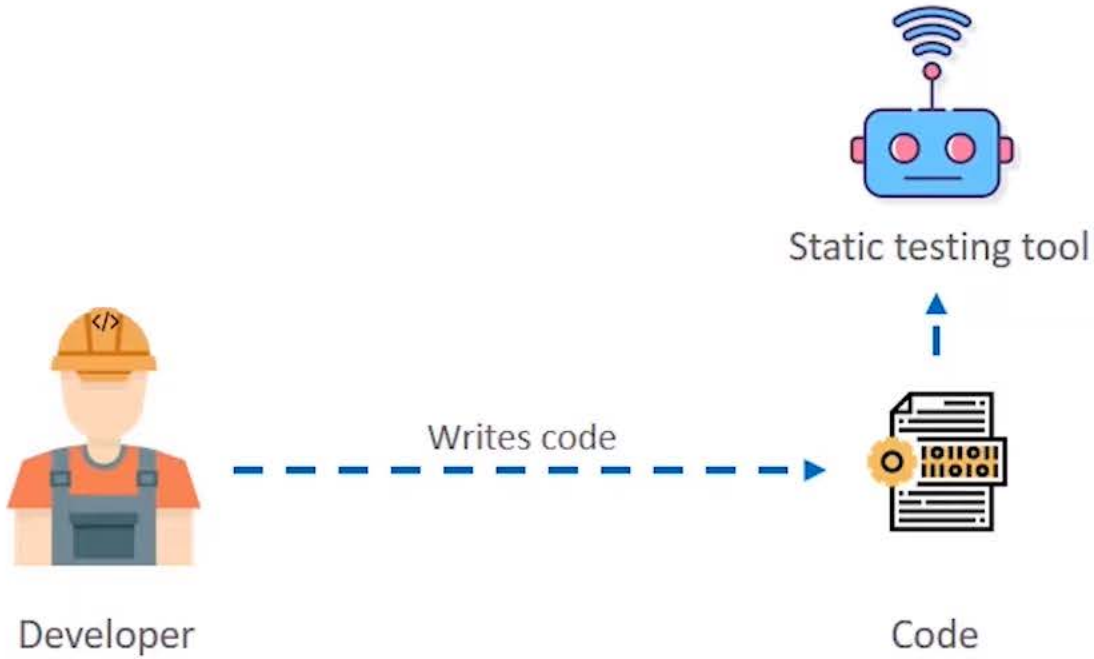
Static Testing

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.



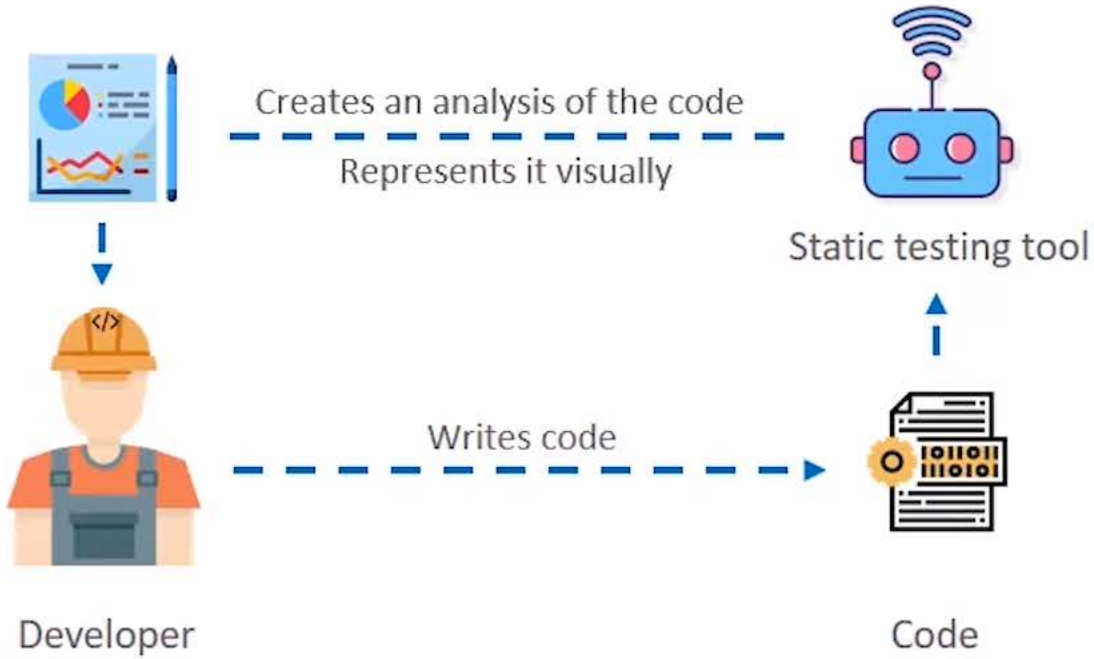
Static Code Analysis

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



Static Code Analysis

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



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

It helps find errors way earlier in development before it goes into production. They are cheap and easy to fix.

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

It helps detect if the code is written in a very complicated matter even though it can be written very easily.

Complex method

Simplified method

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

It helps pick up security errors, which basically means it helps the source code be more secure when it is deployed.

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

Devs may forget to follow best practices specific to a coding language, it can help in solving that issue.

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

It can be a waste of time to regularly ask the testing software to test the code. Therefore to solve such a problem we integrate the static testing tool with Jenkins.

Example Of Static Testing Tools

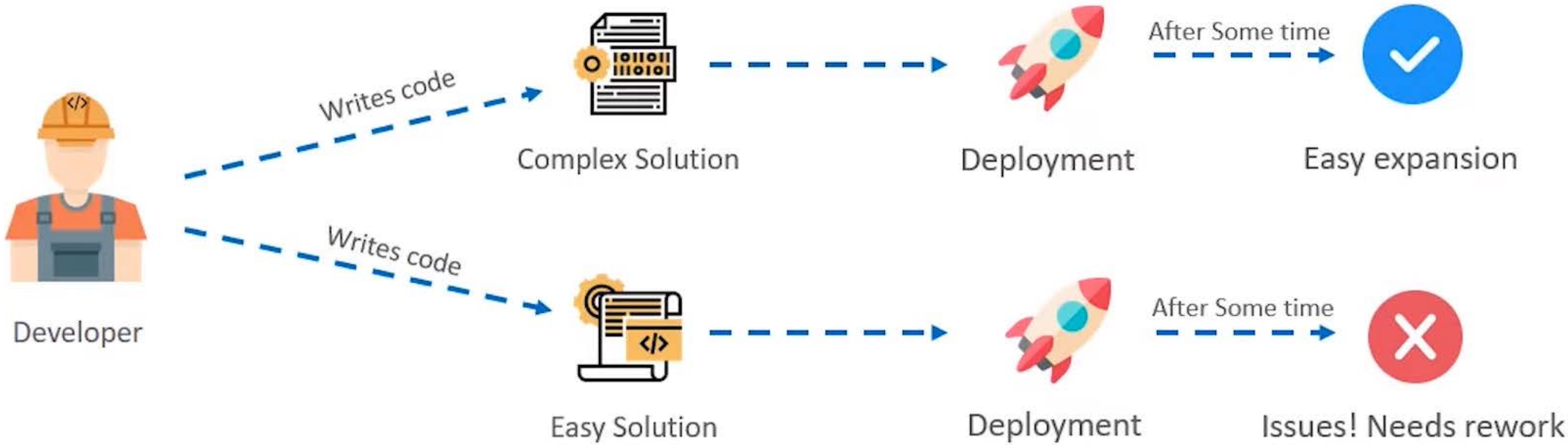
SonarQube

Coverity

Pycharm

Technical Debt

It directly translates as the implied costs for additional rework that can occur if at an early stage an easy but not efficient solution is chosen. In the future the easy code may restrict scalability.



What is SonarQube?

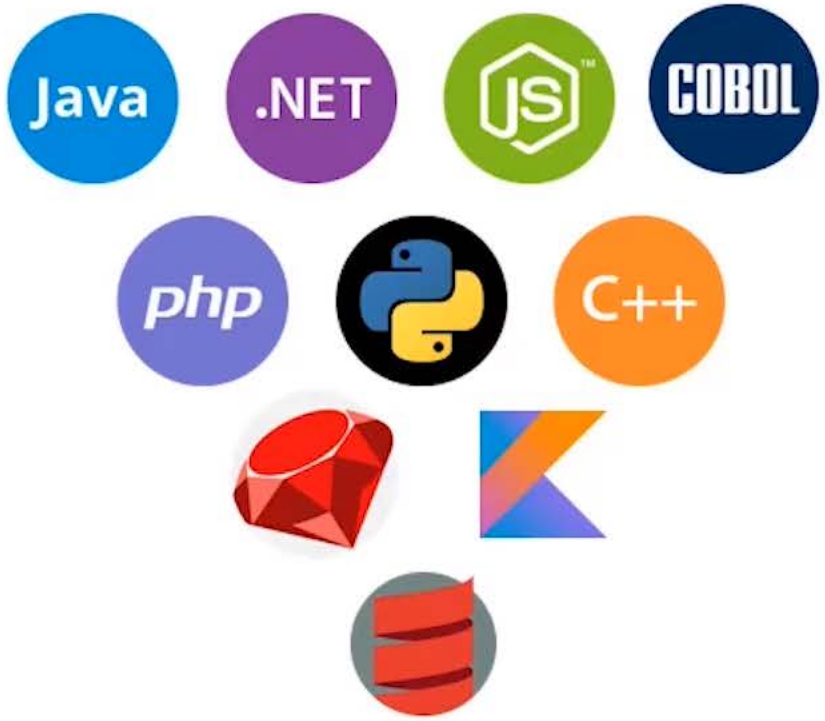
What is SonarQube ?

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

Features of SonarQube ?

It can work with 25 different languages.



Features of SonarQube

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.

Features of SonarQube

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

Features of SonarQube

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.

Features of SonarQube

Tricky Issues

Detect Bugs

Code Smells

Security Vulnerability

Activate Rules Needed

Execution Path

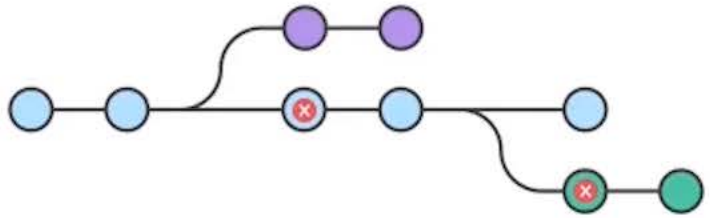
You can create and maintain different sets of rules that are specific to particular projects, these are known as Quality Profiles

Features of SonarQube

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.



Venkat
Corporate Trainer & Motivational Speaker