UNIT TESTING

A quick intododuction of unit testing in projects

ABOUT ME

Javier Martínez Temiño
jmartinez@sqs.es

Working at SQS as testing engineer.
Worked in projects for critical sectors like health or railway.
Unit testing, testing automation, risk and requirements definition & validation.

OBJECTIVES OF THE TRAINING

- What is unit testing? When is recommendable to use unit testing?
- How to program unit tests?
- What tools are available for unit testing?

QUESTION

What is the level of knowledge you have about unit testing?

A. No knowledge neither experience
B. I have studied some unit testing procedures but never worked on it.
C. I have worked on unit testing.
D. I am a certified tester, with high experience on unit testing.

TABLE OF CONTENTS

01

WHAT IS UNIT TESTING?

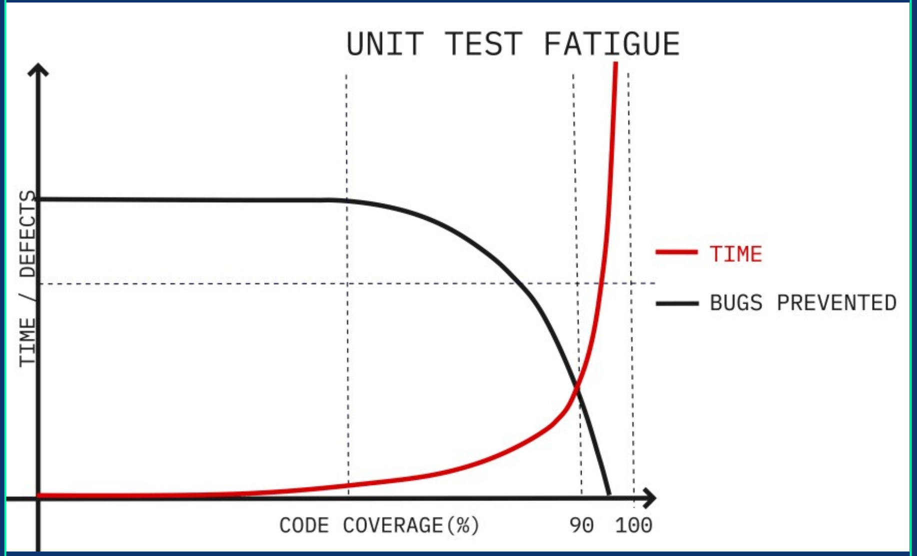
What is it and when is it recommendable using it?

WHAT IS UNIT TESTING?

- Lowest level of testing during software development.
- Most precise.
- Testing functions in isolation. It’s behaviour against it expected behaviour.
- Coding process
- A program that exercises your application’s components

COST OF UNIT TESTING

- Testing fatigue.
- The coverage comes from comparing the tested code against the source code.
- Significant cost in implementation, low cost in execution.



GLOSSARY OF TERMS

STUB/MOCK

Both concepts are similar.
They are the process of simulating a function by naming the call statement, and defining the output expected under a set of input variables.

- A stub only simulates a function
- A mock gives also information about how the function was used

COVERAGE

The coverage of a project is the amount of code tested against the full code. There are different ways of measuring this value:

- Branch coverage
- Statement coverage

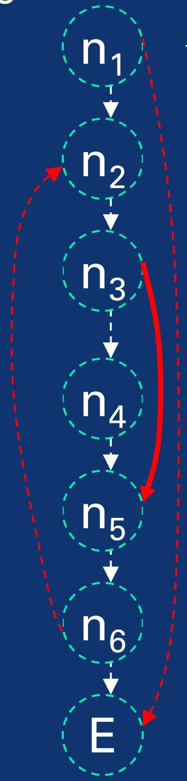
GLOSSARY OF TERMS

Branch coverage

For 100% branch coverage each branch must be executed at least once.

Branch coverage is calculated by dividing the already tested branches by the total branches in the subject under test.

$$C_{\text{branch}} = \frac{\text{Number of executed branches}}{\text{Number of branches}}$$



```
while(nCounter<Max)
{

  nCounter++;


  if(cKey == 'Y')

      {
           sText[nIndex}=cKey;
      }

  nIndex++;


}
```

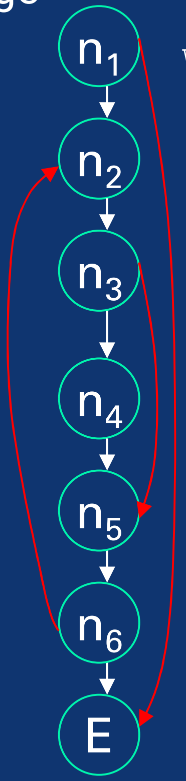
GLOSSARY OF TERMS

Statement coverage

For 100% statement coverage each statement must be executed at least once.

Statement coverage is calculated by dividing the already tested statements by the total statements in the subject under test.

$$C_{statement} = \frac{\text{Number of executed statements}}{\text{Number of statements}}$$



```
while(nCounter<Max)
{

   nCounter++;


   if(cKey == 'Y')

        {
            sText[nIndex}=cKey;
        }

   nIndex++;


}
```

QUESTION

Identify the number of tests required to achieve:

1. 100% statement coverage.
2. 100% branch coverage.

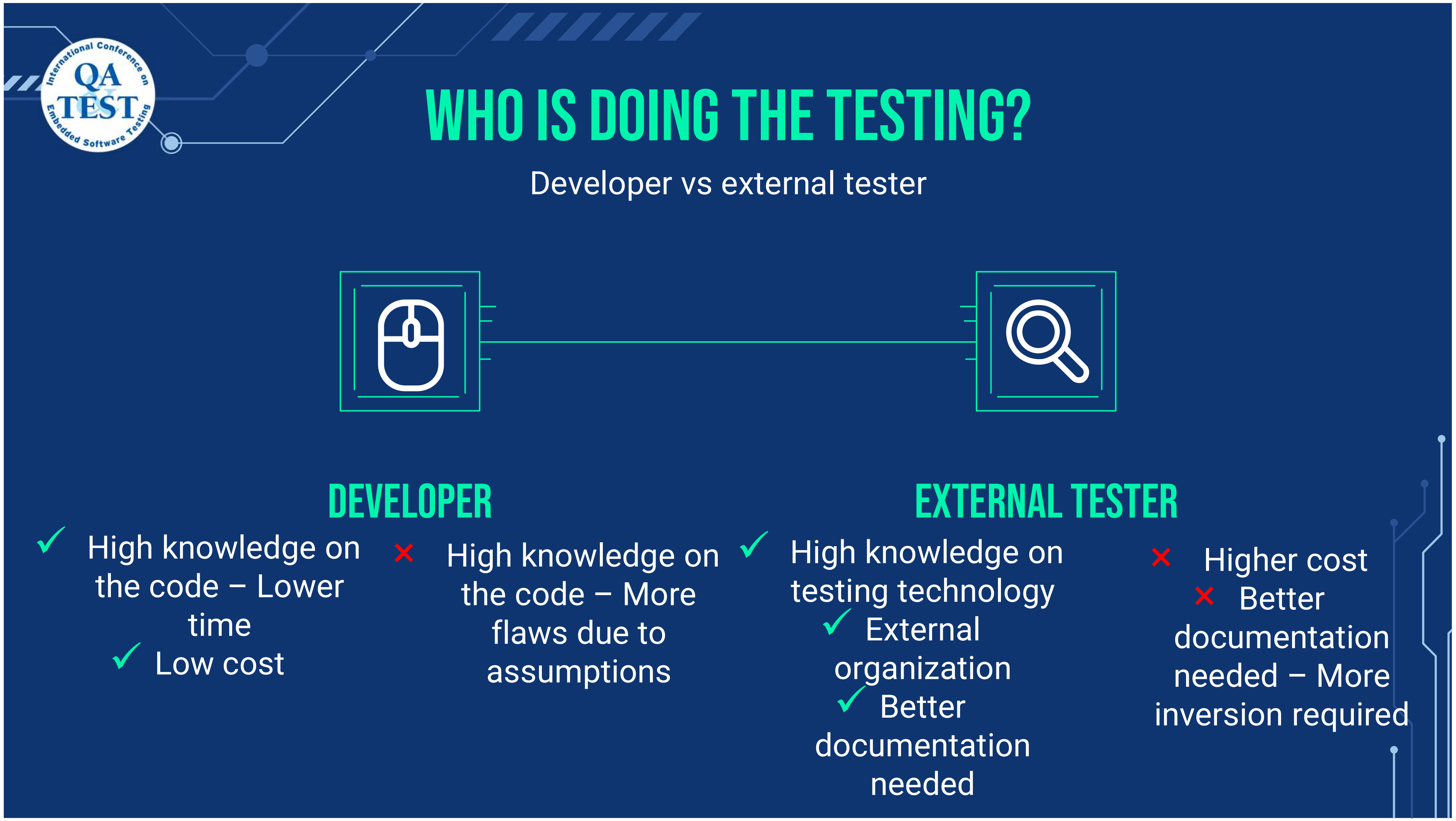
```
READ HUSBANDAGE
READ WIFEAGE
IF HUSBANDAGE>65
        PRINT "Husband retired"
ELSE
        PRINT "Husband not retired"
END IF
IF WIFEAGE>65
        PRINT "Wife retired"
ELSE
        PRINT "Wife not retired"
END IF
```

Statement Coverage ____

Branch Coverage ____

WHEN IS UNIT TESTING RECOMMENDED?

- Certification processes
- When regression testing is used
- Implementing algorithms
- When doing CI/CD development
- Always



WHO IS DOING THE TESTING?

Developer vs external tester

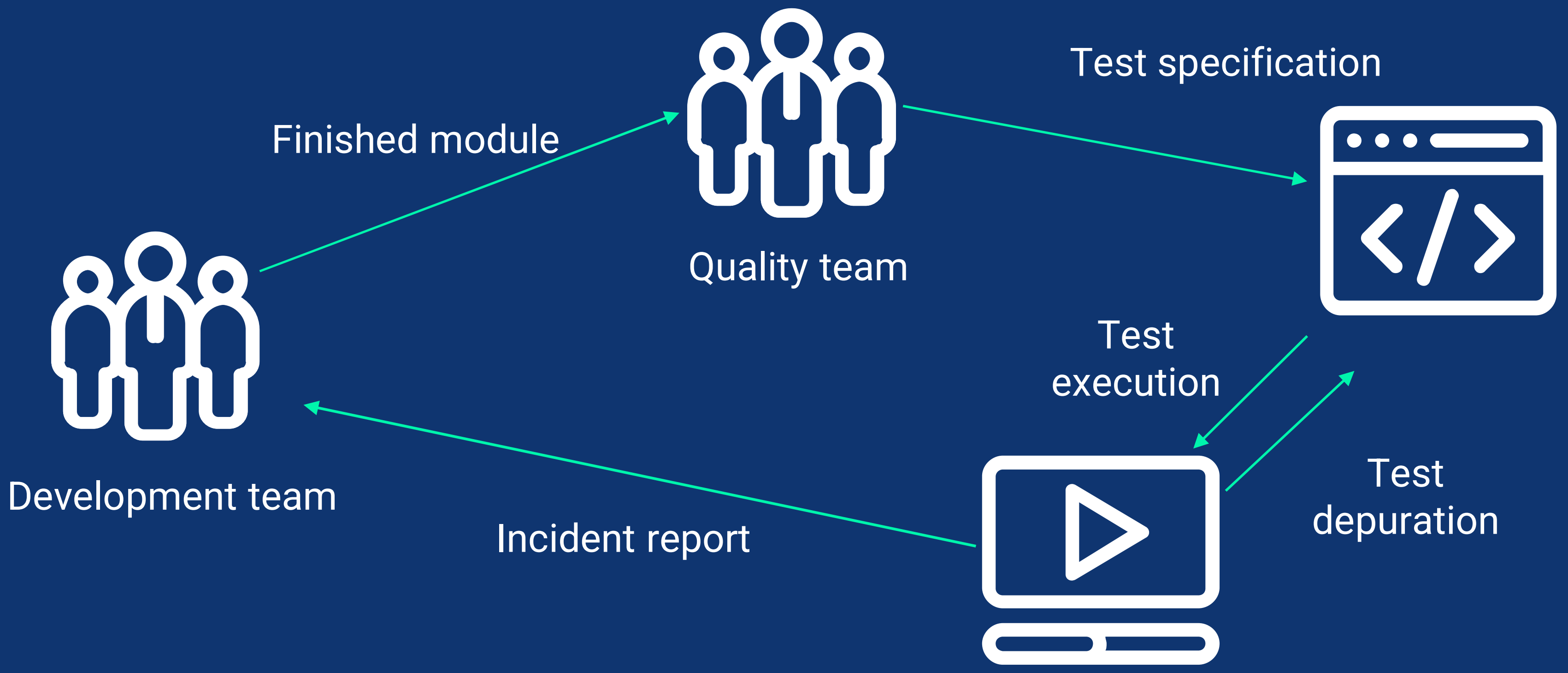
DEVELOPER

- ✓ High knowledge on the code – Lower time
- ✓ Low cost
- ✗ High knowledge on the code – More flaws due to assumptions

EXTERNAL TESTER

- ✓ High knowledge on testing technology
- ✓ External organization
- ✓ Better documentation needed
- ✗ Higher cost
- ✗ Better documentation needed – More inversion required

UNIT TESTING WORK CYCLE



02

HOW TO PROGRAM UNIT TESTS?

The need of software working correctly

BEFORE PROGRAMMING

THE CODE MUST BE DOCUMENTED

THE CODE MUST BE TESTABLE

SELECT TESTING STRATEGY

THE CODE MUST BE DOCUMENTED

The tests are developed based on the documentation

.

All of the functions and structures of the code must have an explanation of its funcionality

THE CODE MUST BE TESTABLE

- Not every code is testable.
- There is not a single quantify way to know if a code is testable
- Code with high complex functions cannot be tested
- A good way to evaluate the code testability is the cyclomatic complexity
- Other ways are the use of global/static functions/variables

TESTING STRATEGY

Order in which TC are done and what they cover.
Level of coverage expected.
Not everything needs to be tested.
Types of test cases development:

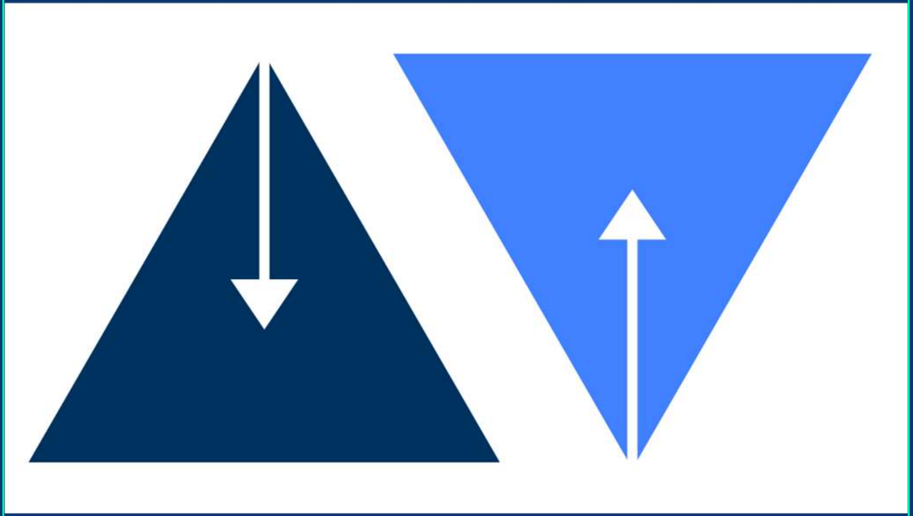
- Bottom up / Top down
- TDD

BOTTOM UP/ TOP DOWN

Reference to the TC creation strategy

- Bottom Up – Start testing from the lower-level modules and then the higher-level ones.
- Top Down – Start testing the higher-level modules and then the lower ones.

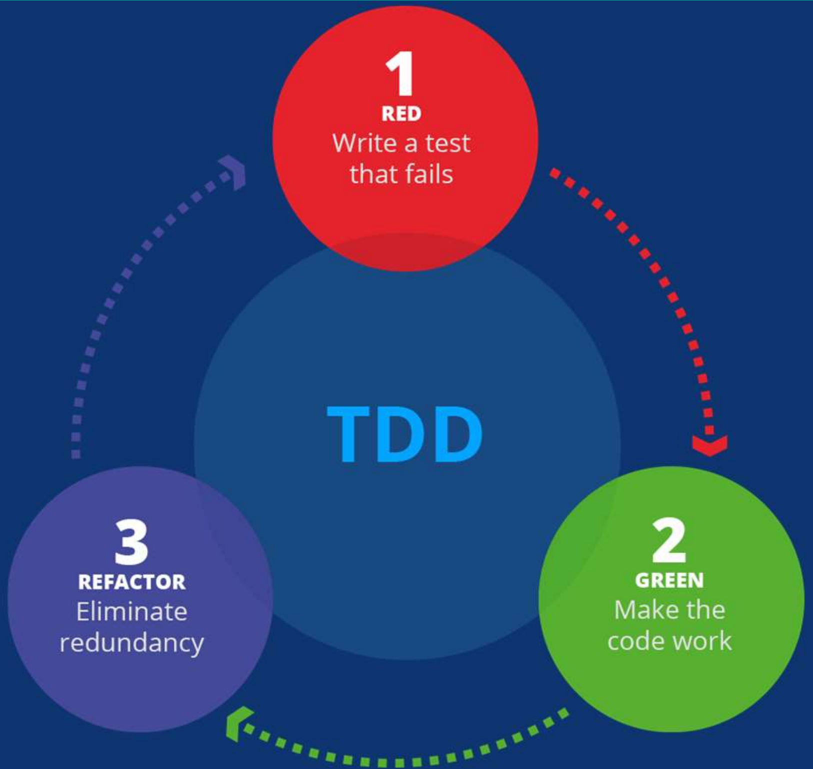
After module testing integration testing is done



TEST DRIVEN DEVELOPMENT (TDD)

Full development strategy.

- First develop a test that fails or doesn’t even compile as there is no function.
- Develop a function that passes the created test cases
- Refactor



LET'S PROGRAM UNIT TEST CASES

PREPARATION OF DATA

- Preparation of input variables
- Stub/mock external dependencies

EXECUTION OF FUNCTION

- Execution of the subject under test

VALIDATION OF RESULTS

- Validation of the output variables against expected results

INTEGRATION WITH EXTERNAL TOOLS

A common practice is to integrate an external code analyzer tool to have further information about the project's test procedure.
This can help to have more information about the coverage done until now, repeated segments of the code or other useful data.
One example of a tool like this can be Sonarqube.

03

TOOLS AVAILABLE FOR UNIT TESTING

The need of software working correctly

WHAT IS A UNIT TEST FRAMEWORK?

- Support structure with the objective of developing test cases and other test-driven functionalities.
- Different levels of complexity, functionalities and price.
- TC depending on the framework can be developed with coding or with a visual interface.

UNIT TESTING FRAMEWORKS

STANDALONE FRAMEWORKS

- Junit - Java
- Nunit - .Net
- SimpleTest – PHP
- Typemock – C++ & .Net
- Cantata – C & C++
- Karma – Javascript

EMBEDDED SYSTEMS

- Vector Cast
- Unity
- Embunit

04

UNIT TEST PROGRAM DEMO

Demo of a program of unit tests
for a java class

THANKS!

Do you have any questions?

jmartinez@sqs.es
www.sqs.es

NEXT TRAINING SESSION

Load and performance testing
21/10/2021 11:00
With Cesar Muñoz