TEST AUTOMATION FUNDAMENTALS

How to automate software test processes

International Conference on QA & TEST Embedded Software Testing

SQS SOFTWARE QUALITY SYSTEMS

ABOUT US

JAVIER MARTÍNEZ

Validation Engineer with background in the Automation area.

jmartinez@sqs.es

CESAR MUÑOZ

Testing specialist, with years of experience in this area.

cmunoz@sqs.es

TABLE OF CONTENTS

TABLE OF CONTENTS

01

INTRODUCTION

What is test automation

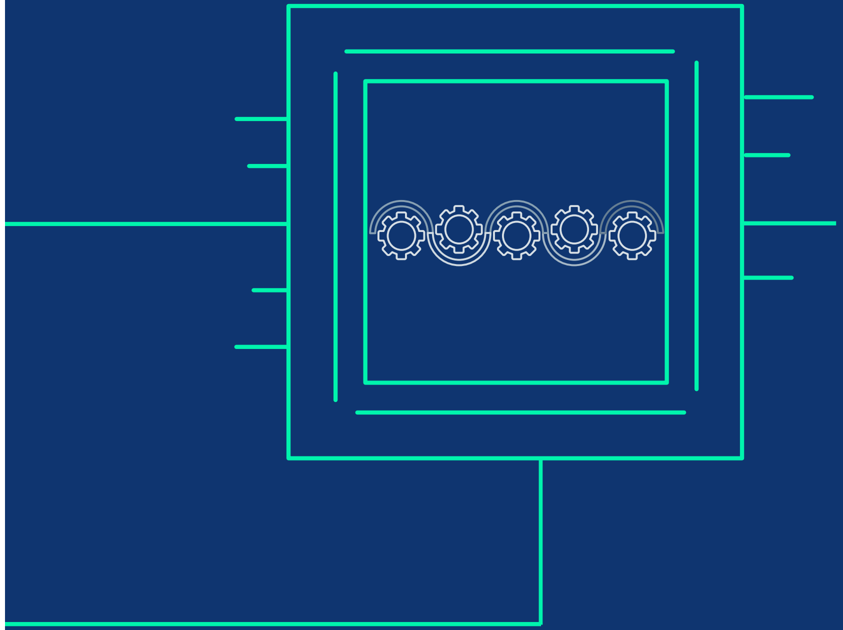
INTRODUCTION

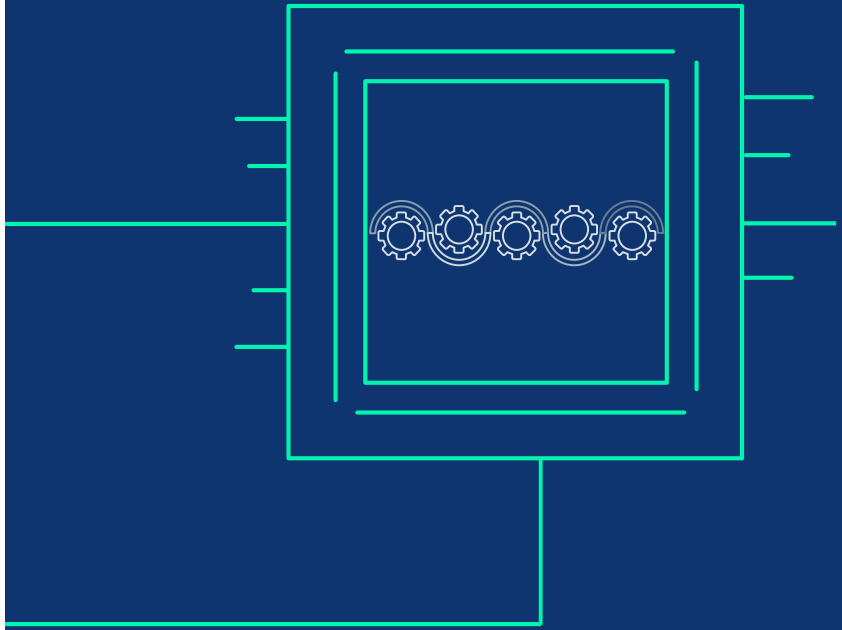
Some test cases are re-executed a lot of times.

Other test cases require a lot of time to complete.

Conclusion: testing could require too much time.

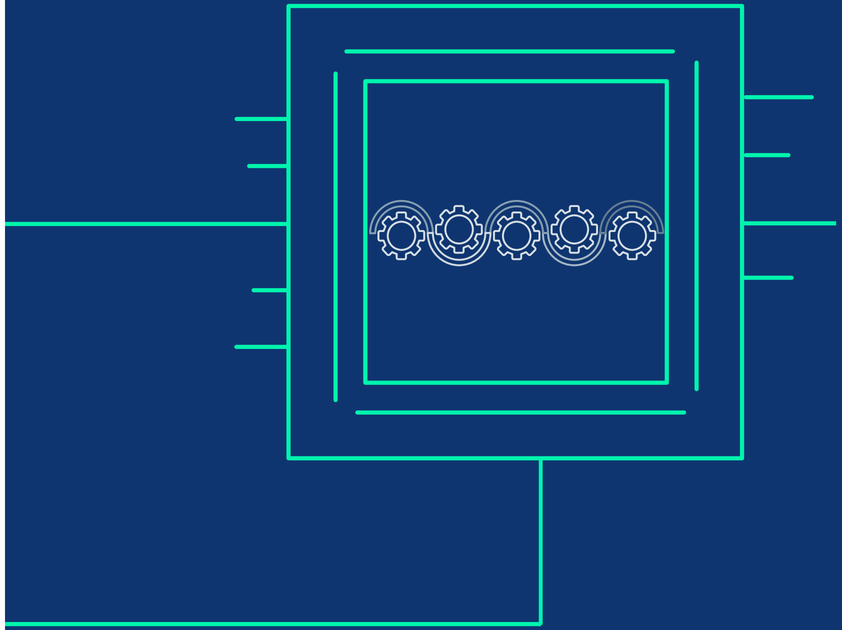
Solution: test automation.





INTRODUCTION

Software test automation is a software testing technique that uses automated testing software tools to create and execute a test case suite



INTRODUCTION

This involves the test configuration/programming, running tests automatically, managing test data, and analyzing results to improve software quality.

INTRODUCTION

Which is your knowledge about test automation?

- I have done test automation
- I have studied about test automation
- No knowledge

02
TEST CASES SELECTION

How to select the test cases to automate

TEST CASES SELECTION

- Not all the test cases are suitable for automation
- A realistic target could be between 30% and 50%.
- The maximum would be 70%.

TEST CASES SELECTION

- The first test cases to automate are the repetitive tasks which demands a lot of time.
- Do not automate test cases which only will be executed one time.
- Compare the effort to automate with the effort to test manually.
- The scripts usually requires updating, so, compare also the effort to update with the effort to test manually.

TEST CASES SELECTION

What should we automate?

- Applications with frequent updates
- Critical applications for the business
- Complex applications
- Applications to be tested on multiple platforms or environments

TEST CASES SELECTION

What should we automate?

- Applications with concurrent users
- Modules used by several applications
- Applications with high costs to fix bugs
- Test cases with special data requirements

TEST CASES SELECTION

We have 3 test cases:
1) It will only be executed once
2) A test case which will be executed more than 10 times
3) A functionality with elements which changes every day
Which test cases will you automate?

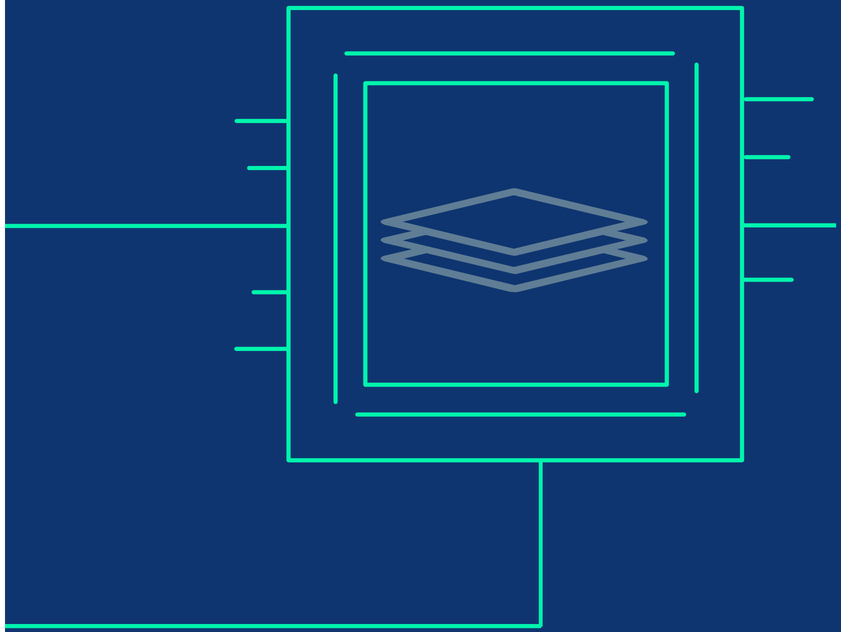
A) ALL
B) 1 and 3
C) 2
D) None

International Conference on QA & TEST Embedded Software Testing

03

ORGANIZATION OF THE AUTOMATION PROCESS

Versions, CI, incidence management, documentation



VERSION CONTROL

Relate the stored test version with its corresponding application version.

Define a policy to create and name new releases:

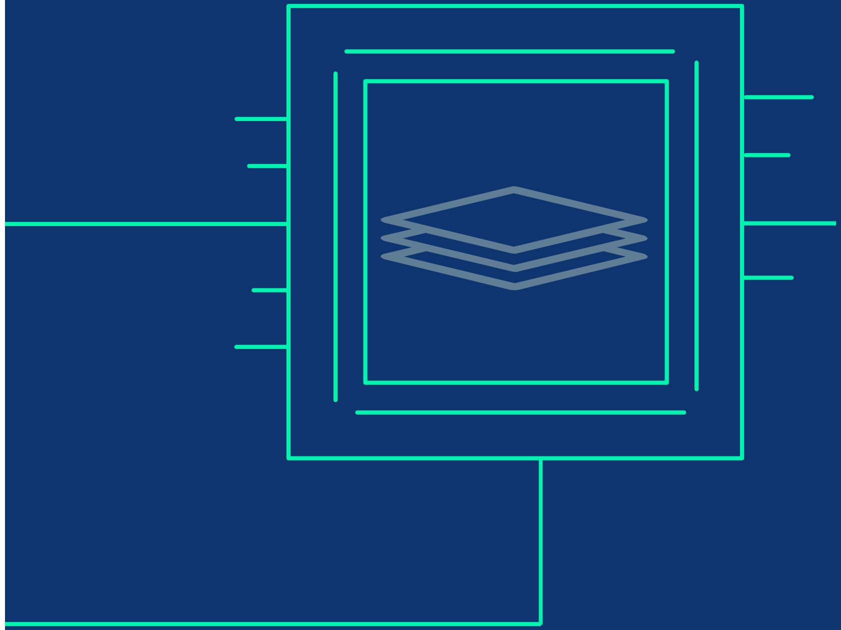
- Nomenclature
- Regularity
- Communication flow

VERSION CONTROL

There are software to implement the version control. Allow accessing to the different versions.

Distinguished version control tools are:

- Git
- Subversion (SVN, TortoiseSVN)
- CVS



CONTINUOUS INTEGRATION

Also known as CI.

The developers send their changes frequently to a central repository. So, the changes from several developers are integrated in one project.

CONTINUOUS INTEGRATION

The automation will be applied to these compilations and can be performed with Selenium or Appium for instance.

Distinguished CI tools are Jenkins, CruiseControl.

INCIDENCE MANAGEMENT

When the automated test detects a failure, first, it has to be described:

- Which point of the test case failed?
- When did it occur?
- Environment?
- Criticality?

INCIDENCE MANAGEMENT

Then, the incidence can be assigned to the group or person who has to treat it.

The incidences have a status:

- New
- Assigned
- Resolved, etc.

INCIDENCE MANAGEMENT

Well known tools which implement incidence management Jira, BMC Remedy ITSM.

Quality Center (ALM) also implements defect management.

Specific tool: Bugzilla

INCIDENCE MANAGEMENT

These are 2 detected fails:
1) When the button “New” is pressed, the screen only shows the message “Deleted the last register”.
2) The field “Address” is shown with very small characters.
What “Criticality” will you select (“High”, “Medium”, or “Low”)?
A) “High” for 1, “Low” for 2
B) “High” for 1, “Medium” for 2
C) “Low” for 1, “High” for 2

CODE DOCUMENTATION

A documentation strategy should be implemented.

This strategy will indicate how to add comments to the code.

The test specifications can be included in these comments.

CODE DOCUMENTATION

Different applications generate documentation from the code comments.

Tools for automated documentation:

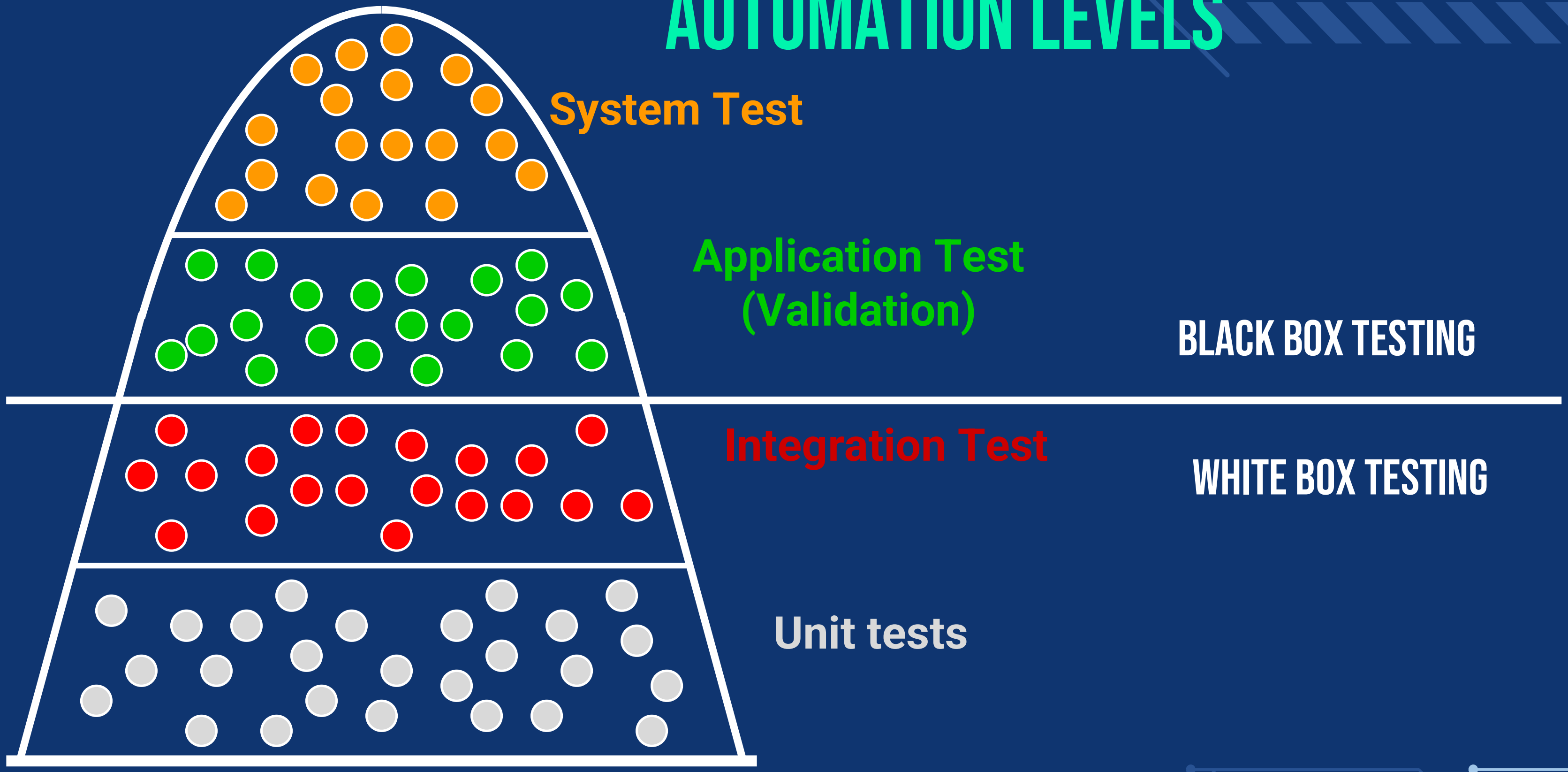
- GhostDoc(C, VB, JavaScript…)
- Doxygen (C, Java, PHP, Python…)
- Pandoc

04

AUTOMATION LEVELS

Unit tests, integration tests, system tests…

AUTOMATION LEVELS



UNIT TESTS

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

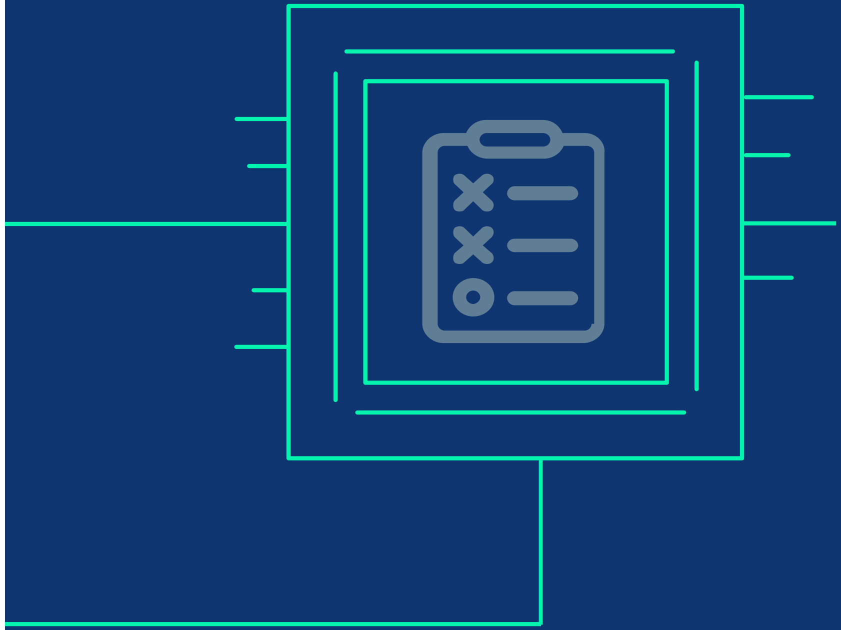
INTEGRATION TESTS

Phase where individual software modules are combined and tested as a group.

INTEGRATION TESTS STRATEGIES

- Bottom up – The lower-level modules are the first integrated. The driver is simulated in order to make the modules work
- Top down – The higher-level modules are the first integrated. The submodules are mocker in order to make the modules work.

APPLICATION TESTS



Process of testing the application in a black box environment with scripts or tools in order to identify functional errors.

It is usually divided in:

- Frontend/ user interface testing
- Backend testing

TYPES OF APPLICATION TESTS

- Smoke and sanity Testing
- Regression Testing
- Acceptance testing
- Functional testing
- Performance testing

SYSTEM TESTS

Level of testing that validates the complete and fully integrated software product.
Series of tests whose purpose is to exercise the full computer-based system.
Some tests are similar to the application tests but in the fully integrated software product.

TYPES OF SYSTEM TESTS

- Load Testing
- Regression Testing
- Recovery testing
- Migration testing
- Functional Testing

AUTOMATION LEVELS

We have the classes “Teacher” and “Subject”. And we have coded a test case which checks if the field “Hours” of a subject is returned correctly.

Which kind of test is this?

A) Unit test
B) Integration test
C) System test
D) Acceptance test

05
AUTOMATION PROCESS STAGES

Setup environment, data organization, execution, check results…

PROJECT DEFINITION

Information about the Project and its requirements is gathered:

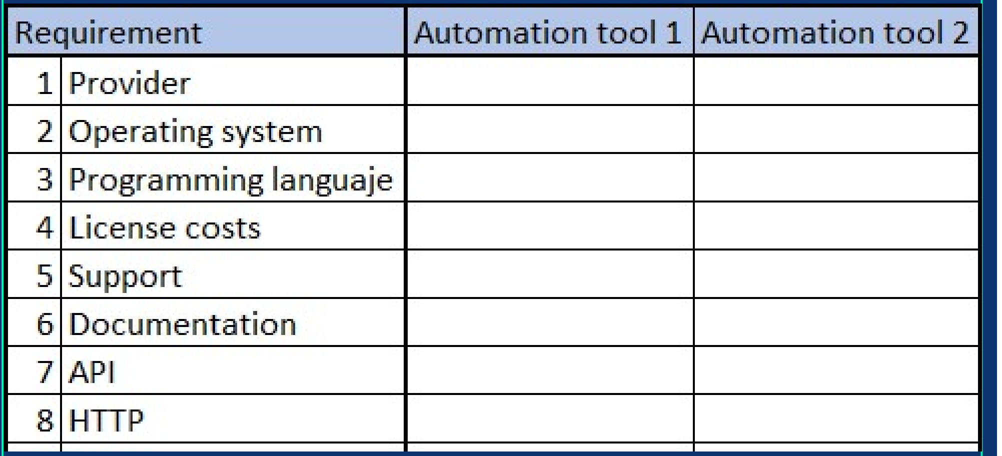
- Budget
- Scope
- Whom contact
- Deliverables
- Automation process estimation
- Prototype

TOOL SELECTION

When selecting the automation tool, we have to consider firstly:

- Application environment
- Application interfaces
- Provider (support, trustworthy)
- Programming language
- Budget

We can make a requirements matrix



Requirement		Automation tool 1	Automation tool 2
1	Provider		
2	Operating system		
3	Programming languaje		
4	License costs		
5	Support		
6	Documentation		
7	API		
8	HTTP		

TEST SYSTEM DESIGN

- Design test environment
- Select and define test cases
- Define how to control the test versions
- Define data and test cases backups
- Define defects managements

ENVIRONMENT CREATION

Create the system architecture:

- Hardware elements (physical or virtual servers…)
- Software elements (database, applications…)
- Dependencies between the different elements

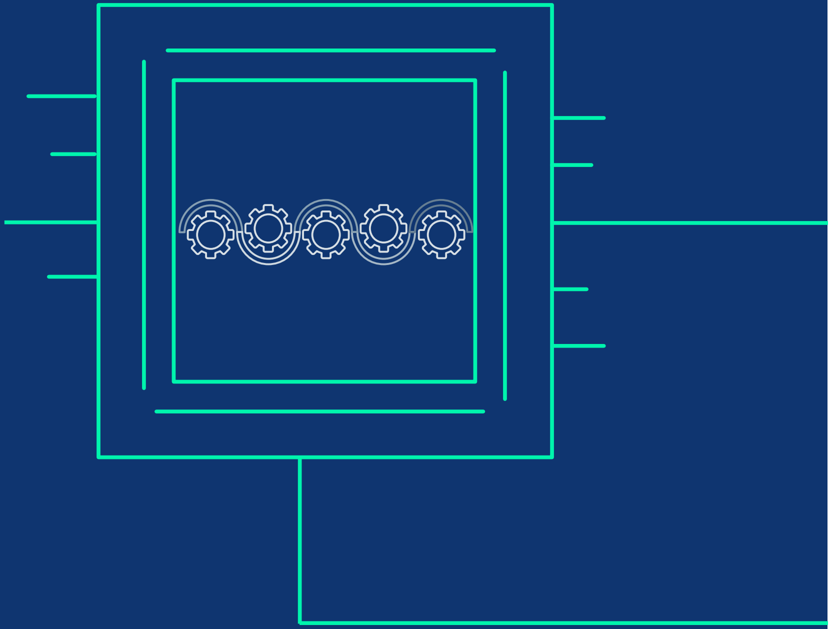
TEST DATA CREATION

Different possibilities:

- Create new (false) data for the test
- Copy data from production
- Use applications to generate data
- Extract from database
- Use the previously existing data from the environment

CREATE AUTOMATIONS

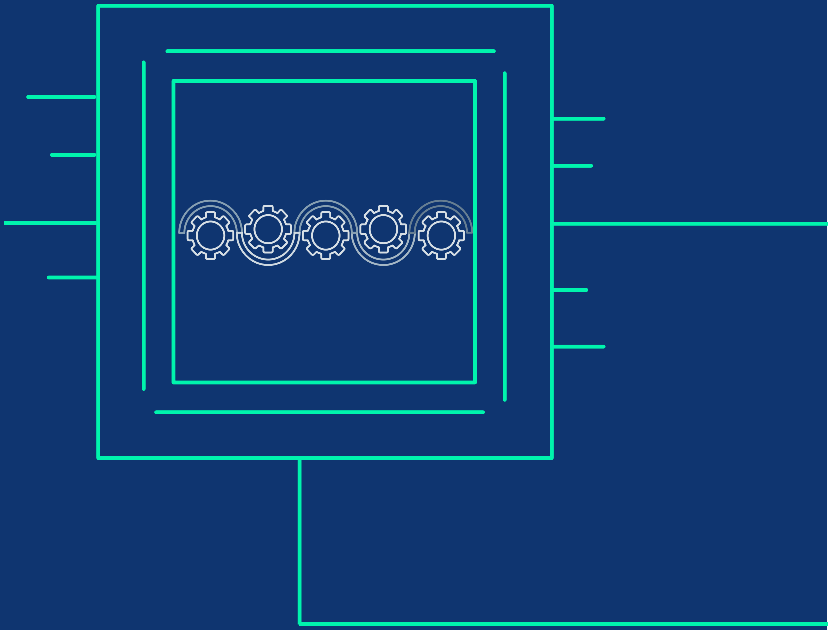
Some applications could require additional elements to allow its automation. For instance, virtual terminal devices for mobile applications.



CREATE AUTOMATIONS

Different possibilities depending on the environment, application, and automation tool:

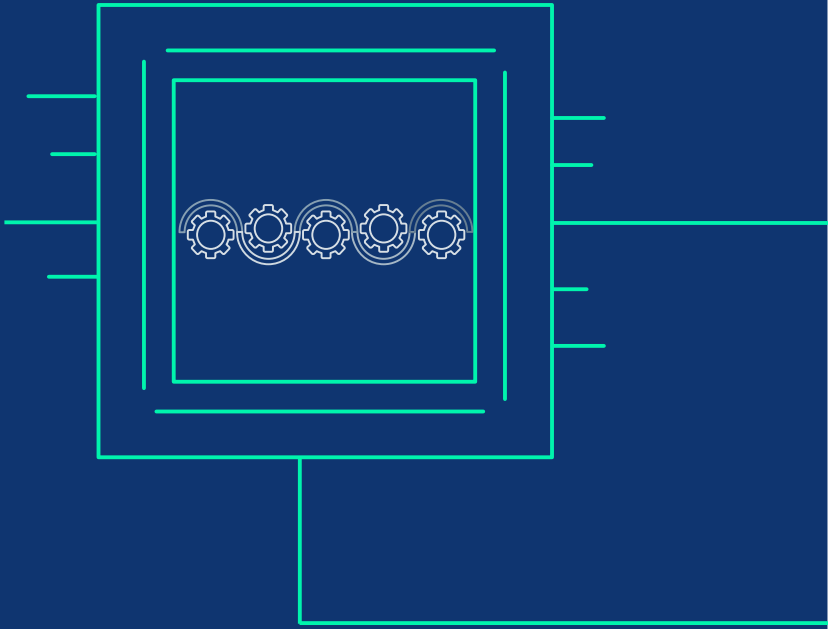
- Program the test case
- Capture the test case when executing manually
- Configure a functionality of the tool we will use to test



CREATE AUTOMATIONS

Usually, the automated tests will require parameters, which will allow us to introduce different data or configurations when executing.

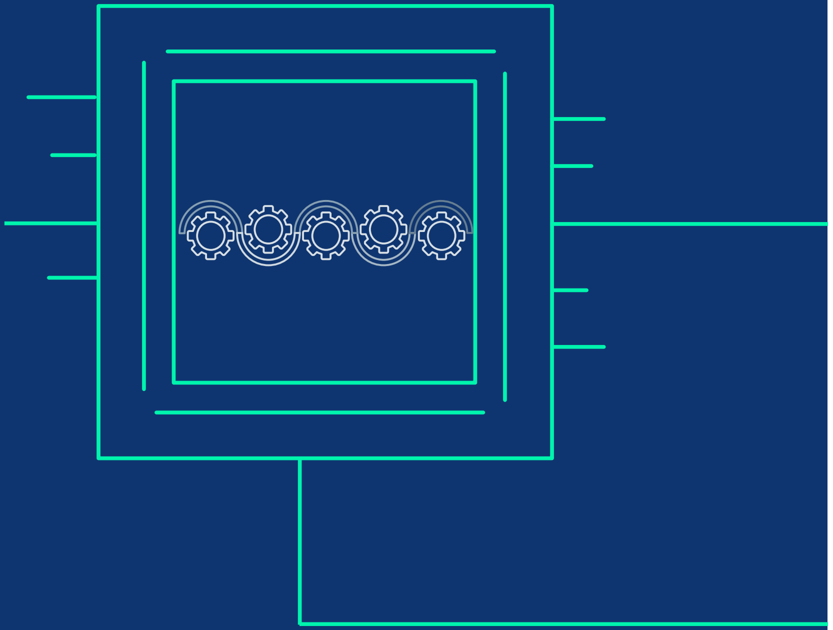
Other kind of parameters could require to catch its value in real time when executing. These are called “correlations”.



CREATE AUTOMATIONS

Finally, the automated test should check its results. If not checked, our automated test could be running without execution errors, but returning wrong results, and we wouldn't be aware.

"Assertions" are a mechanism to compare a result which we select from the execution, with an expected value indicated by us.



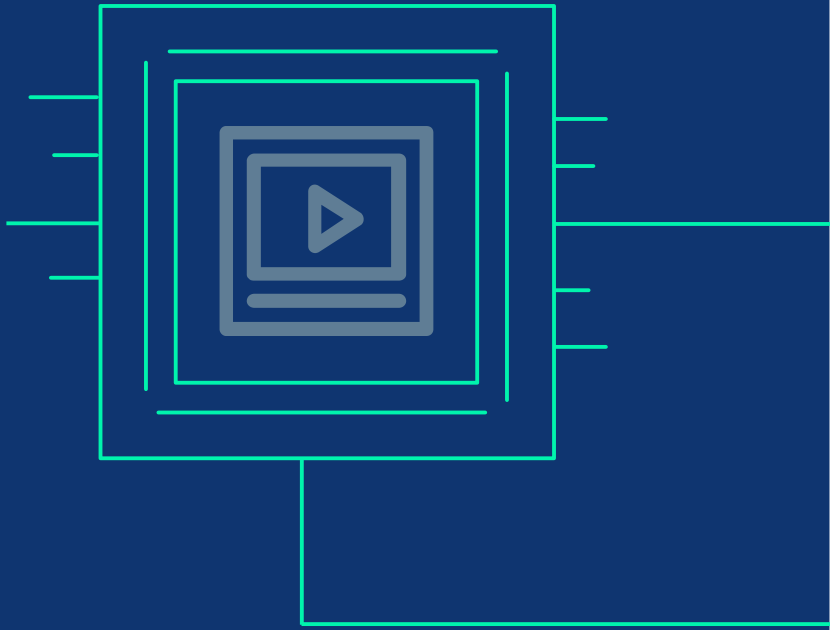
TEST EXECUTION

Before the execution, some elements could require to be started. For instance, if we are testing against a local database, it should be open and running to be accessible.

TEST EXECUTION

We will execute the automations created.

The execution results have to be compared with the expected results. If the comparison hasn’t been automated, we’ll have to do the comparison by our own means.



TEST EXECUTION

Every test case execution has to be registered, including configuration/data used, and result (pass or fail).

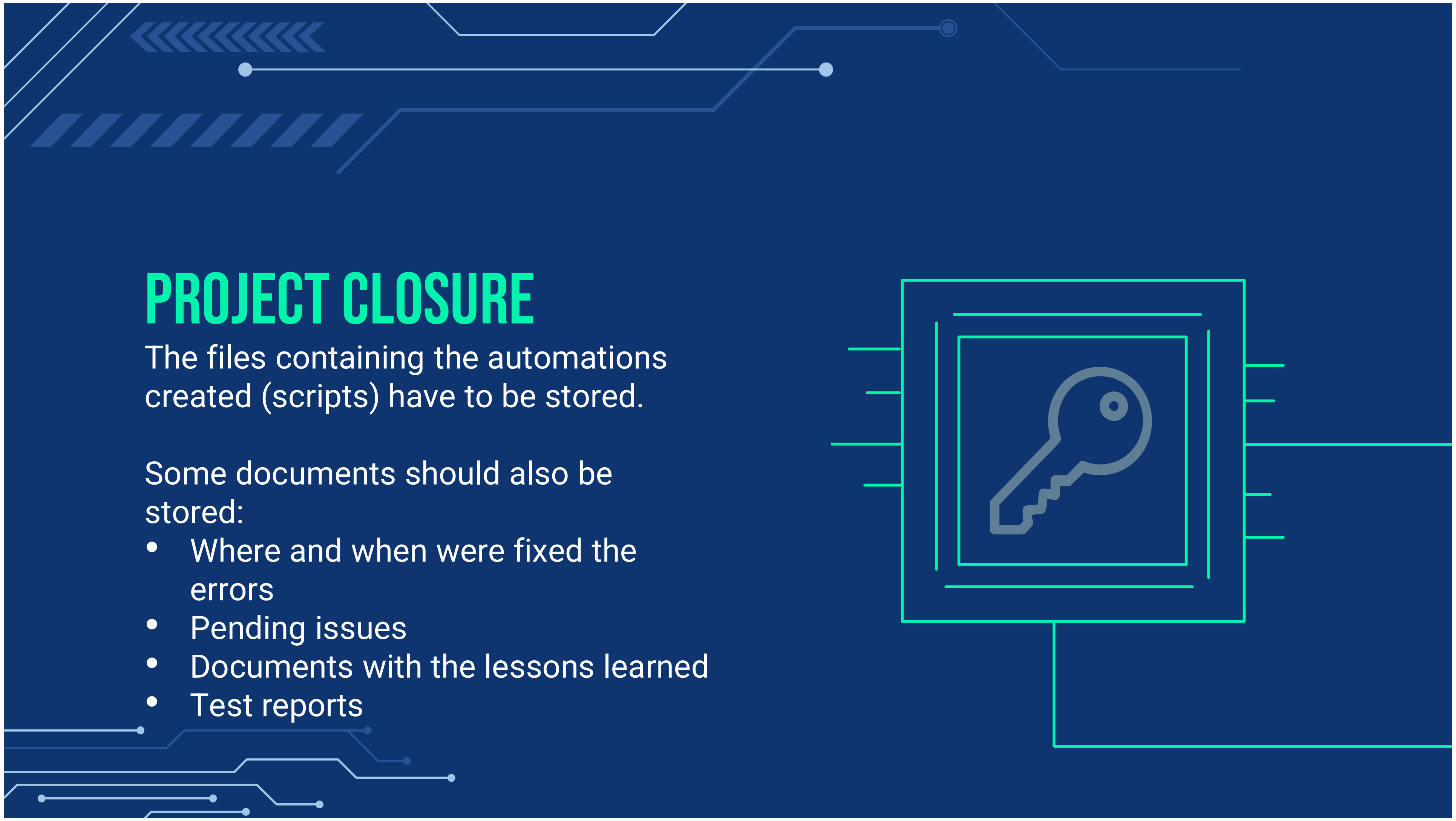
REPORTS

Some tools, after the execution, have the option to create a report with the results.

Anyway, a report with the fails should be delivered to the development team in order to fix them.

EXIT CRITERIA

Once the desired coverage level has been reached, and executed with the desired success rate, the testing process can be finished.

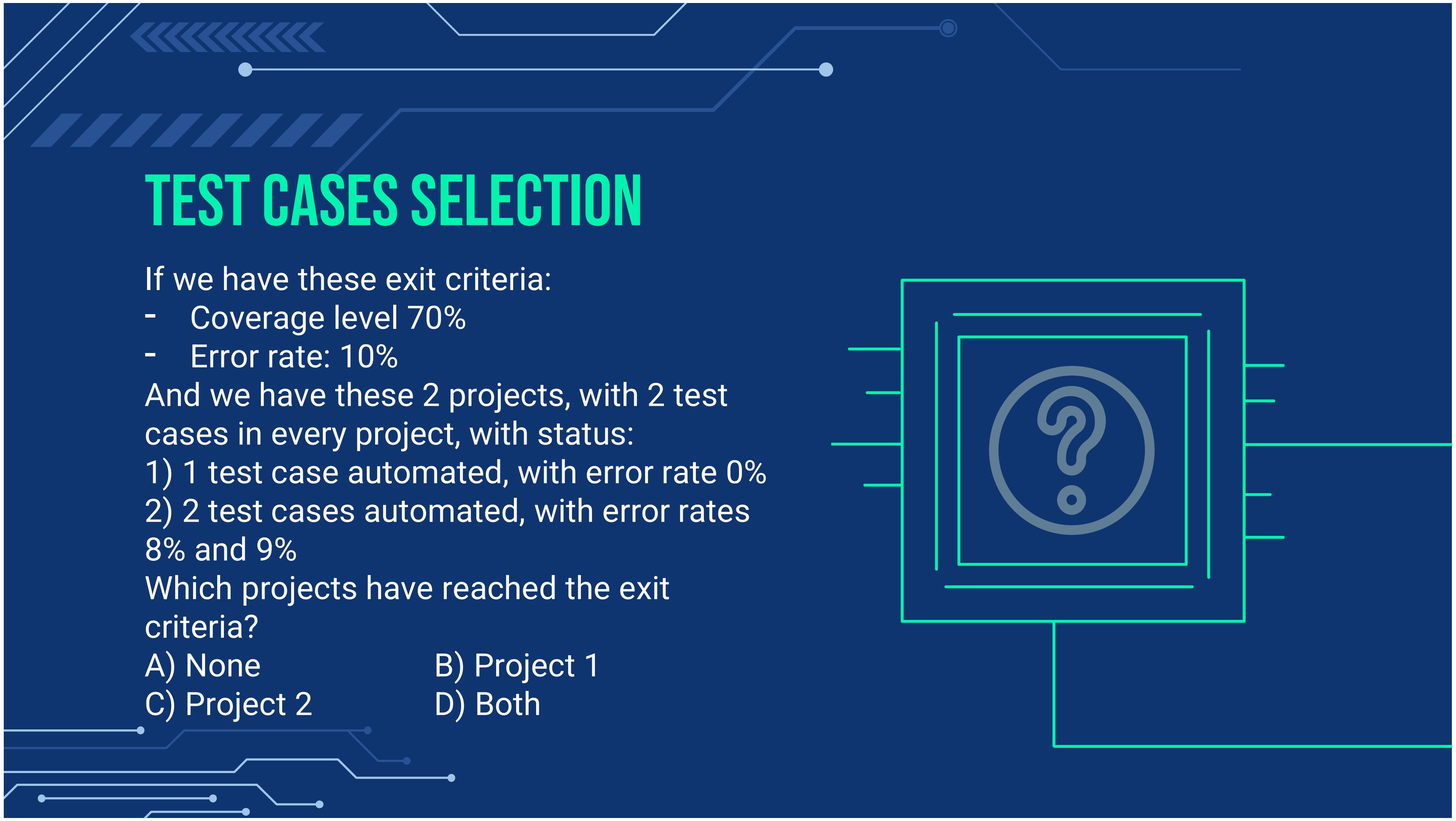


PROJECT CLOSURE

The files containing the automations created (scripts) have to be stored.

Some documents should also be stored:

- Where and when were fixed the errors
- Pending issues
- Documents with the lessons learned
- Test reports



TEST CASES SELECTION

If we have these exit criteria:

- Coverage level 70%
- Error rate: 10%

And we have these 2 projects, with 2 test cases in every project, with status:

1) 1 test case automated, with error rate 0%
2) 2 test cases automated, with error rates 8% and 9%

Which projects have reached the exit criteria?

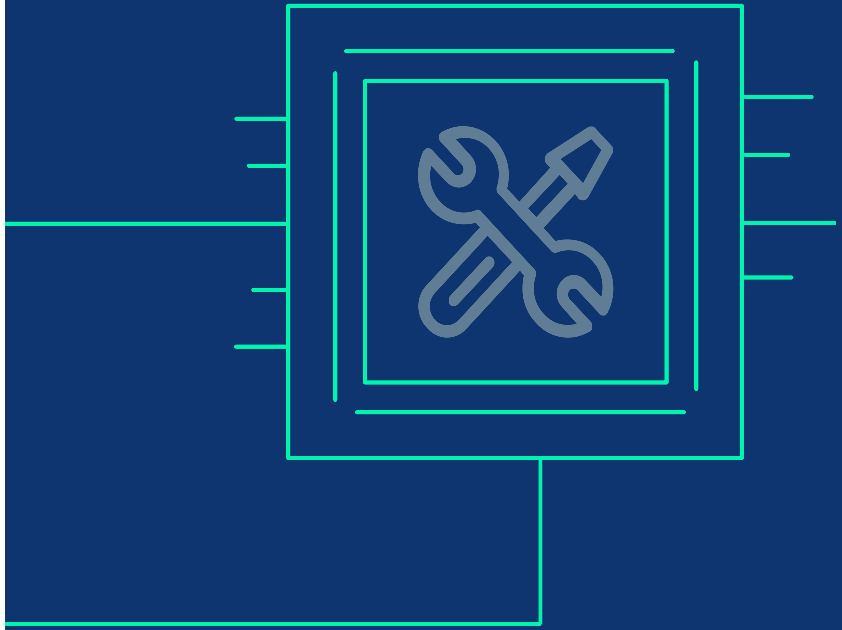
A) None B) Project 1

C) Project 2 D) Both

06

TOOLS

Main automation tools: Selenium, UFT, JUnit, Sonar



SELENIUM

Maybe the most known automation tool.

It is a free automation tools suite:
- Selenium WebDriver
- Selenium IDE
- Selenium Grid

SELENIUM IDE

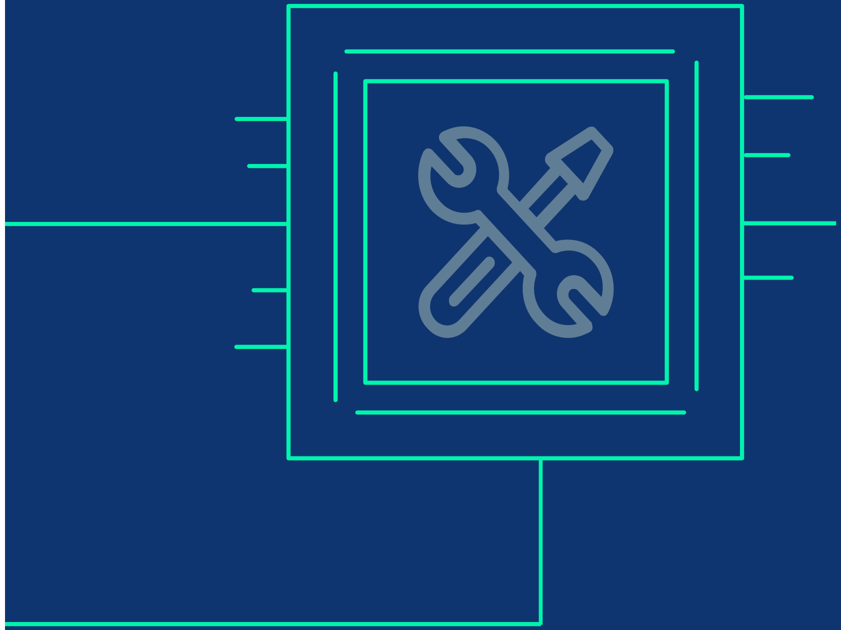
It allows scripts recording.

The script is shown as a table.
Every step is a file with 3 columns:

- Command: action to perform
- Target: web element to apply the action
- Value: optional value to send to the element

SELENIUM GRID

To scale by distributing and running tests on several machines and manage multiple environments from a central point, making it easy to run the tests against a vast combination of browsers/OS.



SELENIUM WEBDRIVER

It is an API. It allows the creation of robust tests.

Automated tests based on browser.

Different programming languages supported: Java, Phyton…

Different browsers supported: Chrome, Firefox…

UFT

HP sold it to Microfocus.

It can automate different Windows applications, not only web applications.

Its scripts must be programmed on Visual Basic Script.

Easy integration with ALM.

JUNIT

It is an open source testing framework for Java applications.

It allows not only unit tests, but also integration tests.

The test class should include “Test” in the class name.

JUNIT

Three main kind of annotations for the test methods:

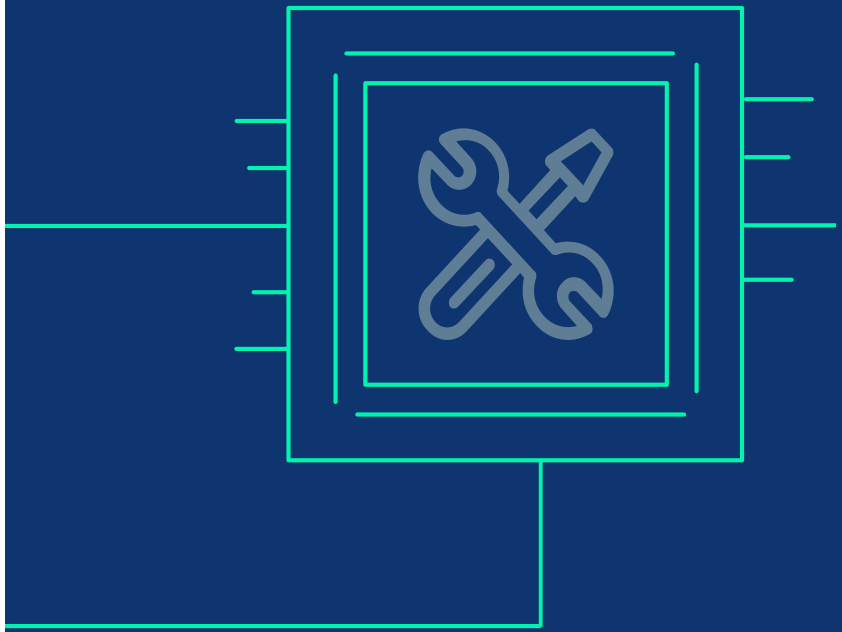
@Before: executed before a new test case is going to be executed

@Test: test case and its validations

@After: executed after a test case has been executed

JUNIT

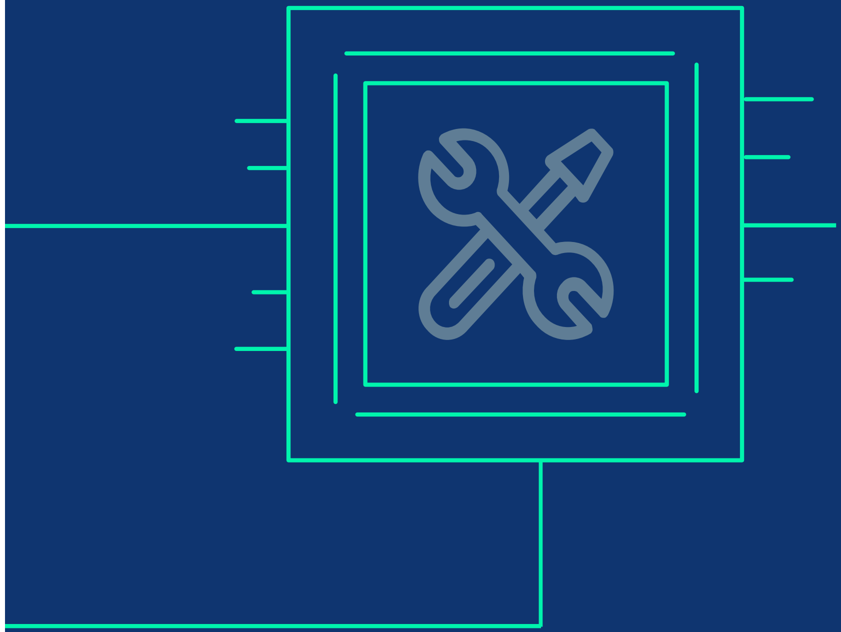
The test methods will include "assertions" once the test case has been completed. This is, to check a condition compliance to get the result of the test: pass or fail.



SONARQUBE

SonarQube is an open-source product for code quality static analysis to detect:

- Bugs
- Duplicated code
- Not accessible code
- Design problems
- Security vulnerabilities
- Codification standards



SONARQUBE

It provides free analysis for fifteen programming languages:

- Java	- XML	- JavaScript
- C#	- Scala	- Kotlin
- CSS	- Go	- Python
- Flex	- PHP	- VB.NET
- HTML	- Ruby	- TypeScript

More languages available, but not for free

SILK TEST

Automated functional and regression tests for web, mobile, and business software applications.

It works with different technologies: Mobile (iOS, Android); .NET (WinForms, WPF); Java (Swing, SWT); DOM; IE; Firefox; Chrome; Edge; Safari; SAP Windows GUI.

SILK TEST

Silk test offers 4 products:

- Silk Test Workbench allows to test automation tests at a visual level, using VB.Net as scripting language.
- Silk Test Classic uses the domain specific language “4Test” to automate command sequences. Its an object programming language.

SILK TEST

- Silk4J was created to automate in Eclipse using Java as scripting language.
- Silk4Net was created to automate in Visual Studio using VB or C# as scripting language.
- Currently Silk4J and Silk4Net has been fused into “UFT Developer”

JMETER

Apache JMeter is usually used for performance testing. But this requires a previous test case automation. In fact, it is sometimes used simply for automated functional testing, without load generating.

JMETER

JMeter is a Java desktop application (but, when testing performance it's recommended to launch the tests from the command line).

JMETER

JMeter saves every request as an object, which can be configurated.

Also, additional components can be added to the script or to the requests, like:

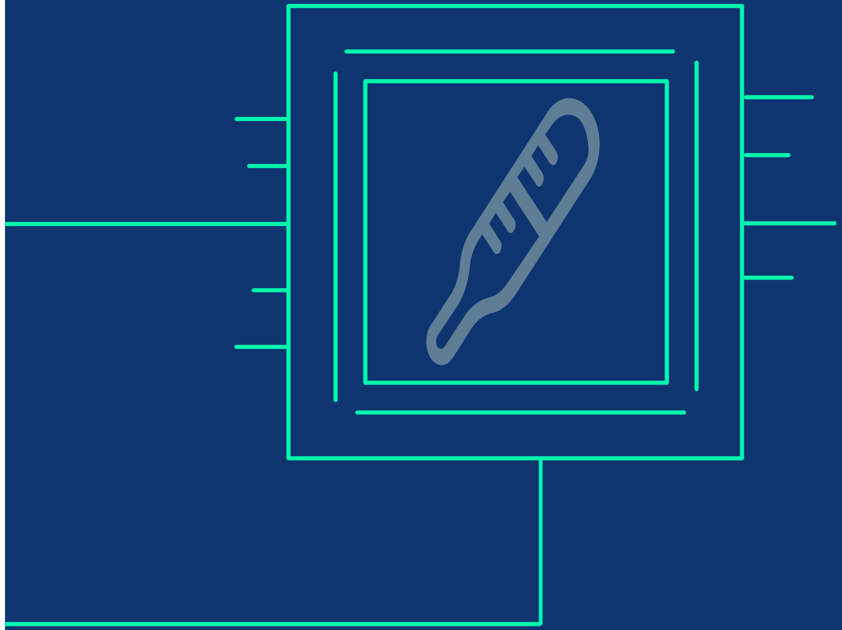
- Graphics
- Cookies management
- Assertions
- Result reports

LOADRUNNER

Like JMeter, it is a performance testing tool, which requires to automate the test cases previously (LoadRunner Virtual user generator).

The generated scripts runs also on the monitoring tool BSM.

It works with different protocols and sources.



LOADRUNNER

When capturing test cases, the source can be a browser or a client desktop application.

Depending on the selected protocol, the scripts will test the GUI (for instance the TruClient protocol), or the network communications (for instance the HTTP, HTML or Web services protocols)

LOADRUNNER

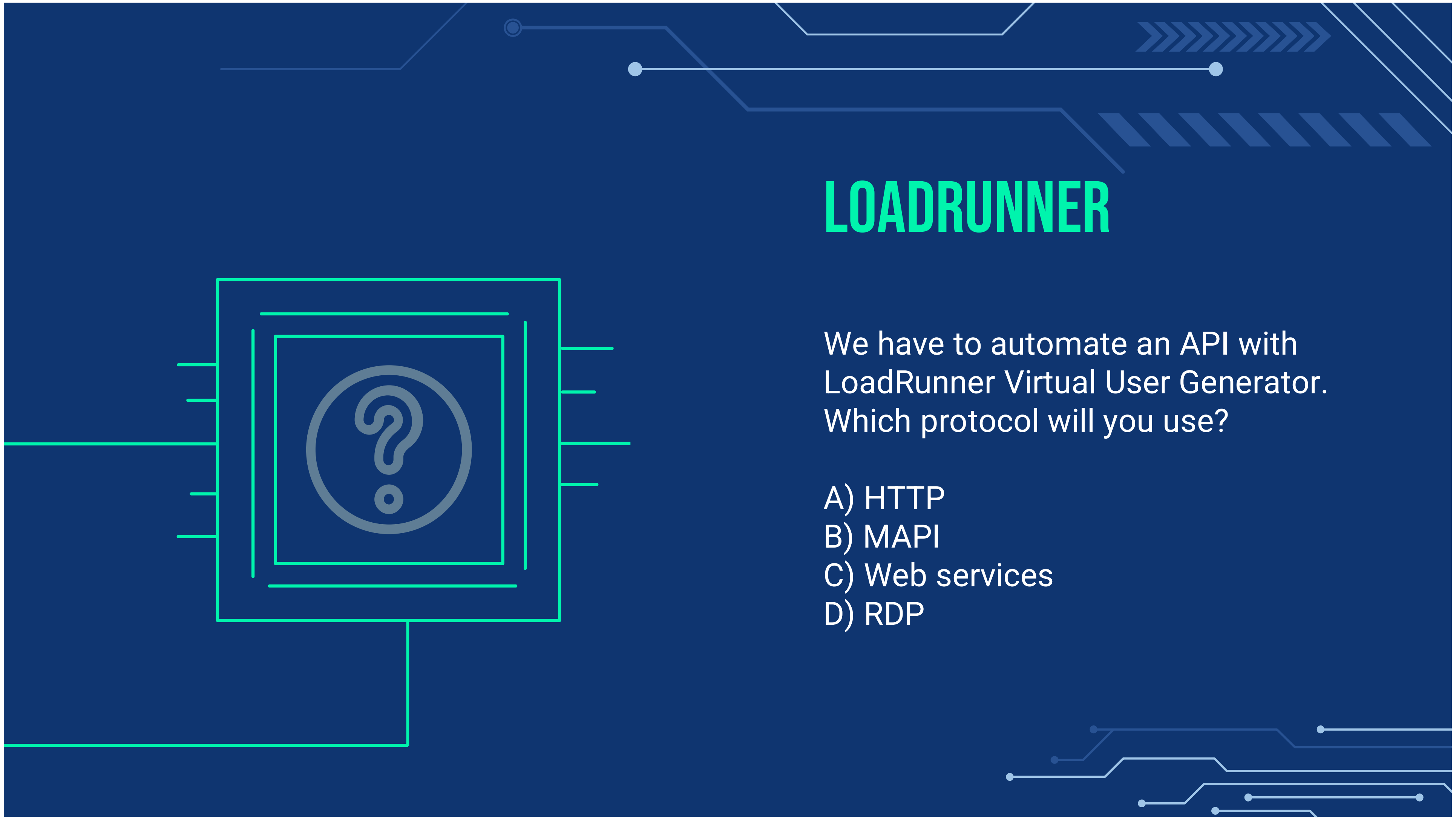
Some protocols:

- HTTP/HTML: reproduces the request from the client application.
- Web services: SOA requests
- MAPI: for Microsoft Exchange
- RDP: connection to a remote machine
- TruClient: reproduces the user actions on a web interface (GUI)

LOADRUNNER

HTTP and HTML protocols are the most used. LoadRunner Virtual User Generator (VUGen) captures the navigation done by the user, and saves it a as C or Java script.

C is the most used language for the LoadRunner scripts.



LOADRUNNER

We have to automate an API with LoadRunner Virtual User Generator. Which protocol will you use?

A) HTTP
B) MAPI
C) Web services
D) RDP