ATTAFI YOUSRA

Selenium Certified Tester

Part 1
Basics of automation

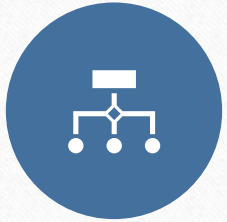
Knowledge levels of learning objectives

01	02	03	04
K1: remember	K2: understand	K3: apply	K4: analyze

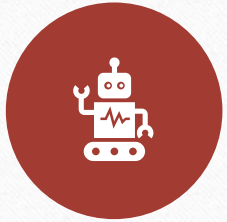
The automation of tests

Objectives of the chapter

STF-1.1 (K2) EXPLAIN THE OBJECTIVES, ADVANTAGES, DISADVANTAGES AND LIMITATIONS OF TEST AUTOMATION



STF-1.2 (K2) UNDERSTANDING THE RELATIONSHIP BETWEEN MANUAL AND AUTOMATED TESTING



STF-1.3 (K2) IDENTIFY TECHNICAL SUCCESS FACTORS FOR A TEST AUTOMATION PROJECT

What is test automation?

As far as we are concerned, the automation of tests is based on execution equipped with functional tests

Automation usually simulates a manual test run by a human

Despite aspects related to automation, analysis, design and implementation are generally manual

Overview of Test Automation

- Automation is the design, creation and maintenance of various levels of untestware:
  - The environment in which the tests will be performed
  - One or more specialized test tools and harnesses
  - Code libraries that provide functionality
  - The test scripts
  - A monitoring and reporting platform to evaluate results

Overview of Test Automation

- Depending on the tools used, the test machines can ensure
  - Monitoring and controlling the execution of tests
  - Creation and deployment of data used in tests
  - Evaluating the success or failure of a test
- Often in a combination of manual and automated processes.

Objectives of Test Automation

- Improve test efficiency by reducing the cost of each test.
- Test aspects other than manual tests.
- Reduce the time required to run the tests.
- Push more tests earlier in the software development life cycle
- Increase the frequency of testing.

Test automation/ Benefits

- Gain in efficiency
- Some tests that cannot be performed manually
- Reduced test run time
- Increased frequency with which certain tests can be performed
- Release of manual testers
- Reduction of errors made by manual testers
- Running tests earlier in the process
- Perform tests outside of normal working hours.
- Increased confidence in the build

Automation of tests/ Technical risks

- The complexity and/or size of automated tests can become very important.
- Software development skills required
- Significant maintenance of tools, environments and test assets may be required.
- It is easy to add technical debt
- Test automation involves all processes and disciplines of software development.
- Risk of losing focus on project risk management.
- The pesticide paradox is increasing
- False positives occur
- Tools tend to have only one common thread

Automation of tests/ Project risks

- Unrealistic expectations from hierarchy
- Short-term thinking that can hinder an automation project
- Organizational maturity is necessary for success
- Some testers are perfectly satisfied with manual tests
- Automated test oracles may differ from manual test oracles
- Not all tests can or should be automated
- Humans find most bugs
- A false sense of security due to the large number of automated tests
- Cooperation with development is necessary, which can create organisational problems.

Manual vs Automated Tests

Early versions of the test automation tools were a complete failure. They sold very well, but rarely worked as advertised

This is partly due to the fact that they failed to perform relevant software tests and simply did not understand the role of the tester in the testing process

The main problem with these tools was incorrect modelling of the tests

- Each test was run identically each time
- The test was greatly underestimated
- The scripts were heavily impacted by the many GUI changes

Context and Likelihood

A manual test script tends to contain information in three columns:

- The first column will contain a summary of the abstract task to be performed.
- The second column tells the tester which data to use to complete the task.
- The third column tells the tester what behavior to expect.

Context and Likelihood

Each task is filtered by the knowledge of the tester, who brings value to the test.

In running a test as in checking the results, the manual tester adds to the test script:

- A context
- A likelihood

that give intelligence to tests.

An automation tool has no **intelligence**. An automated test using an automation tool has limited **context** and **likelihood.**

Additional cost of automated tests

- However, an automated script can have an intelligence built in by the test automation engineer through programming.
- This intelligence leads to an **additional cost in development and** maintenance.
- Because an automated script requires more analysis, more design, more engineering and more maintenance than a manual script, it is necessary to calculate the cost of its creation.
- And, once created, it must be maintained permanently: When the SUT is modified, automated scripts will usually have to be modified together.

Success factors

- A hierarchy formed to understand what is possible and what is not
- A business case for the short, medium and long term.
- A long-term plan aligned with business needs.
- Mature levels of documentation
  - Test automation strategy must be documented
  - Automation architecture and framework must be documented
  - A formal plan must be documented for automation maintenance
- A development team that includes automation
- SUT designed to be testable.
- Good tools for working in the environment and with the SUT(s).
- Good training, including testing and development techniques.
- Well-designed automation architecture and framework
- Automation at the right interface level

Interface levels

- There are several different levels of interface that we can automate for a given SUT:
  - At the GUI level – HMI
  - At the API level
  - At the level of private hooks (APIs that are specific to tests)
  - At the protocol level (HTTP, TCP, etc.)
  - At the service level (SOAP, REST, etc.)

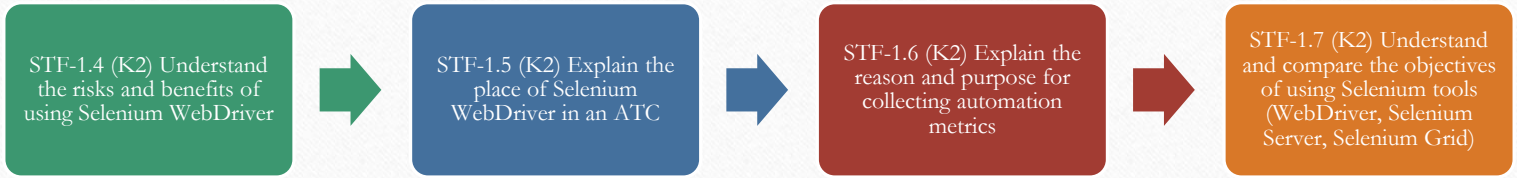
- The lower we are in this list, the more robust/ the less fragile the tests will be

Manual tests vs automated tests: conclusion

- A business case must be made before and during the automation of the tests.
- Often the study will be negative. The reasons for this:
  - There may not be a positive ROI to their automation.
  - The reasoning process of the manual tester may be essential to the proper completion of the test.
  - The testability of the SUT is poor
- In all cases, exploratory tests, injection/foul attack tests and other types of manual tests remain necessary for successful testing.
- You can't automate everything

Automation with Selenium Webdriver

Objectives of the chapter



How Selenium Webdriver works

- Selenium WebDriver works using its APIs (Application Programming Interfaces) to be able to make direct calls to a browser using the native support of each browser for automation.
- Each supported browser works slightly differently.
- Like any tool, using Selenium WebDriver has both benefits and risks.

Advantage of Selenium Webdriver

- Many of the benefits an organization can enjoy are the same as any other automation tool, including:

  - The execution of tests can be uniform and reproducible.
  - Suitable for regression tests
  - Detects defects not detected by API level tests.
  - Lower initial investment
  - Compatibility with different browsers allows you to perform compatibility tests.
  - Support for different programming languages
  - Useful for agile teams.

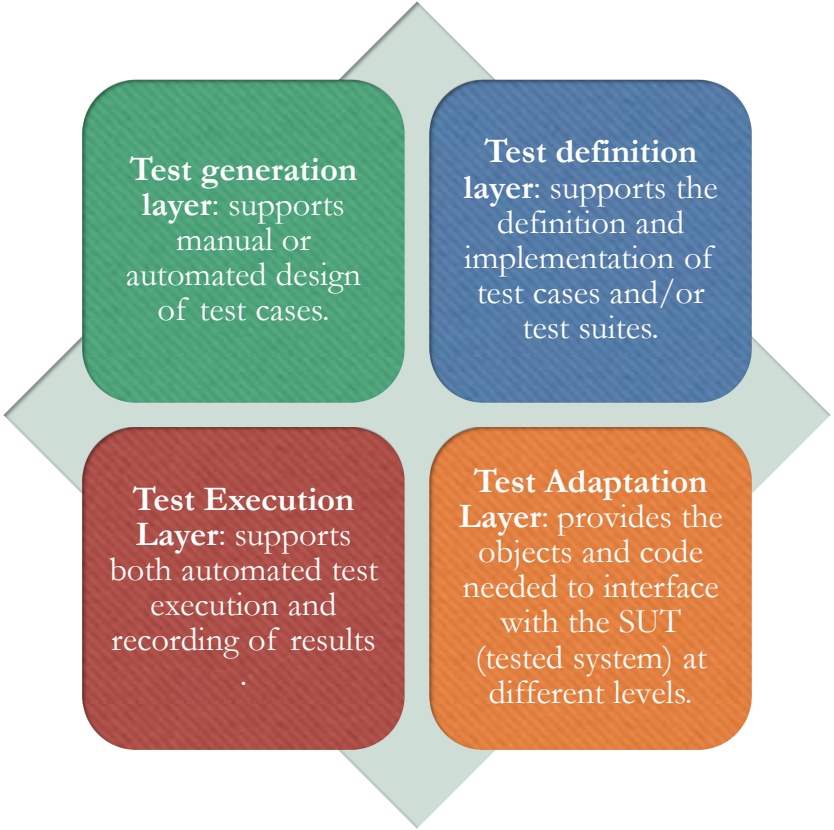
Risks of Selenium Webdriver

- Organizations are often so preoccupied with testing via the GUI that they forget the testing pyramid
- Automation can make the build much longer than expected.
- Changes to the UI have a greater impact on tests than unit level or API tests.
- Manual testers may be more effective at finding bugs than automation.
- Tests that are difficult to automate may be ignored.
- If the web application is more or less stable, automation may not amortize.

Selenium WebDriver in Test Automation Architecture

- The Test Automation Architecture (AAT), as defined by the ISTQB in the Advanced Test Automation syllabus, is a set of layers, services and interfaces of a Test Automation Solution (SAT).
- The AAT consists of four layers (see illustration below):

The 4 layers of the AAT



Selenium in the AAT

- Selenium WebDriver fits into **the test adaptation layer**, providing a programmable way to access SUT via the browser interface.
- The test adaptation layer facilitates the separation between the SUT (and its interfaces) and the tests and test suites we want to run on the SUT.
- This separation allows test cases to be more abstract and disconnected from the tested system.
- So, when the tested system changes, the test adaptation layer, (in this case WebDriver), allows to modify the automated test, which allows it to run the actual concrete test case on the modified SUT interface.

Selenium in the AAT

- The automated script with WebDriver uses the API to communicate between the test and the SUT as follows:
  - The test case requires a task to be performed
  - The script calls an API in WebDriver
  - The API connects to the appropriate object within the SUT
  - The SUT responds as requested (or not, if the test step fails)
  - Success or failure is communicated to the scenario
  - If successful, the next step of the test scenario is called in the script

Languages supported by Selenium

- WebDriver is a programming interface for the development of advanced Selenium scripts using the following programming languages:
  - C#
  - Haskell
  - Java
  - JavaScript
  - Objective C
  - Perl
  - PHP
  - Python
  - R
  - Ruby

Browsers supported by Selenium

- The browsers supported by Selenium (and the component needed to test with) are:
  - Chrome (chromedriver.exe)
  - Internet Explorer (IEDriverServer.exe)
  - Edge (MicrosoftWebDriver.msi)
  - Firefox (geckodriver.exe)
  - Safari (safaridriver)
  - HtmlUnit (HtmlUnit driver)

Automation metrics

Measurements are one of those practices that many test engineers like to ignore or hide, because they are difficult to quantify and justify.

Yet automation tends to be quite costly in terms of human effort and tooling costs.

There is little or no ROI in the very short term; the value comes in the long run. Asking Management for a significant investment in time and resources without a business case (including collecting metrics to provide evidence) is asking them to take our word for it.

Metrics: what to measure?

- The AuT syllabus mentions several areas where metrics can be useful, especially at the level of:
  - Usability
  - Performance
  - Reliability
  - Maintainability

Metric project

- Fixed costs for setting up and running automation
- Regression testing effort that was saved by automation.
- Effort made by the automation team to support automation
- Covered configurations tested
- Number of successful passes between failures.
- Frequent failure patterns of automation
- Number of failures observed in automated tests, compared to the actual failures of the SUT reported by these automated tests

Coverage metrics

- Coverage of instructions and decisions for unit tests
- Interface or data flow coverage for integration tests
- Coverage of requirements, characteristics or risks identified for system tests
- Coverage of tested configurations.
- Cover of user stories
- Coverage of use cases

The Selenium Toolbox

The Selenium ecosystem consists of four tools, each of which plays a different role in test automation:

- Selenium IDE
- Selenium WebDriver
- Selenium Grid
- Selenium Standalone Server

Selenium IDE

- It is an add-on to Chrome and Firefox web browsers.
- Its main function is to record and replay user actions on web pages. It also allows you to insert checkpoints during recording. Saved scripts can be saved as HTML tables or exported to different programming languages.
- The main advantages of Selenium IDE are its simplicity and good quality element locators. Its main drawback is the lack of variables, procedures and control flow instructions.

Selenium WebDriver

- A framework enabling test scripts to control web browsers. It is based on HTTP and has been standardized by W3C.

- Utilizing libraries that implement the WebDriver API for various programming languages allows combining its ability to control web browsers with the power of general-purpose programming languages.

- This enables the use of these language libraries to build complex test automation frameworks, including log capturing, assertion handling, multithreading, and more."

Selenium Grid

- It allows running test scripts on multiple machines with different configurations.

- Selenium Grid enables distributed and simultaneous execution of test cases. The architecture of Selenium Grid is highly flexible, as it can be configured to utilize numerous physical or virtual machines with various combinations of operating systems and web browser versions.

- At the core of Selenium Grid, there is a hub that controls the other nodes and acts as a single point of contact for the test scripts. In most cases, test scripts that execute WebDriver commands do not need to be modified to work on different operating systems or web browsers.

Selenium Standalone Server

- This tool is written in Java and is delivered as a .jar file that implements the functions of hubs and nodes for Selenium Grid. It needs to be started separately (outside of the test scripts) and correctly configured to fulfill its role in the testing environment.

ATTAFI YOUSRA
EXPERT EN TEST LOGICIEL
@ : ATTAFI.YOSRA@HOTMAIL.FR
LINKEDIN : Yousra ATTAFI