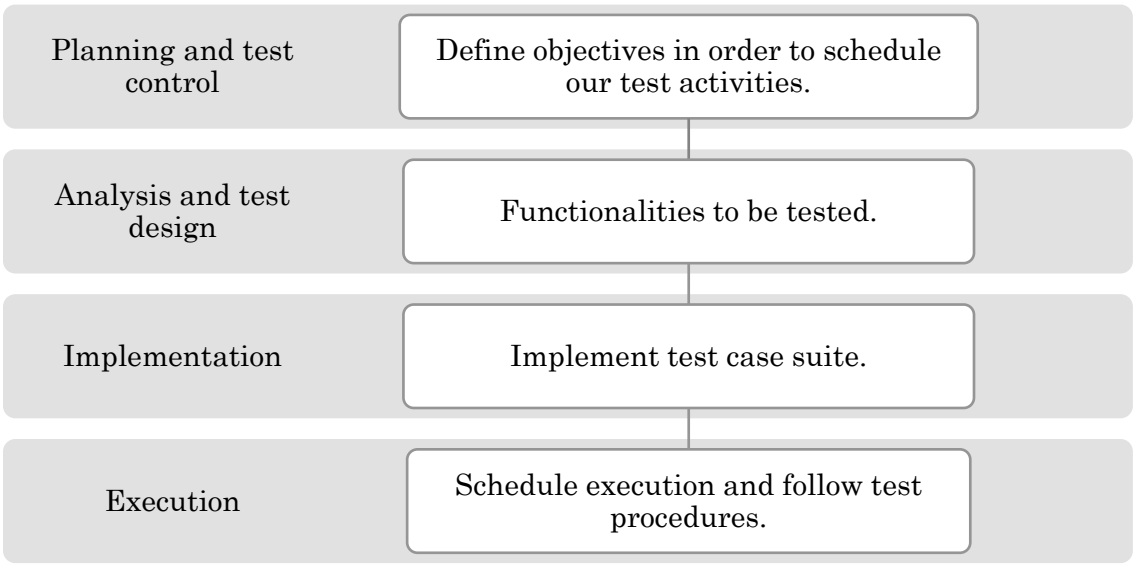
REGRESSION TESTING PLAN

Giovanny De León
Senior QA Engineer

Activities will be divided within the fundamental test process into the following basic steps:

1. Analysis and test design
2. Planning and test control
3. Implementation
4. Execution



Analysis and test design

We start by prioritizing test cases. We must be clear about which area has to be prioritized for testing, that will be our starting point. For the test cases design we must be clear about all the different combinations of suitable data, doing an impact analysis for it. With this we can decide if the test case is accepted or not.

Test case design

Formally, testcases are composed of subdivisions that provide information about the functionality to be tested. In this case, both positive and negative scenarios were considered. And were divided into the following sections to make the test suite structure.

Regression test suite structure

For this automated regression test suite, the following scenarios were added in order to cover and validate CRUD actions can be performed for Computers Database from the UI.

1. Page Elements
    - Page Elements For Main Page Are Displayed
    - Page Elements For Edit Computer Page Are Displayed
    - Page Elements For Add Computer Page Are Displayed

2. Filter Computers By Name
    - Results Are Returned When The Computer Name Exists In The Database
    - No Results Are Returned When The Computer Name Does Not Exists In The Database

3. Update Existing Computer
    - Update An Existing Computer
    - Update An Existing Computer Failed

4. Create New Computer
    - Add A New Computer
    - Add A New Computer Failed

Planning and test control

The requirements are captured and the areas to work with are identified. Once everything is clear, planning decision can be made. We need to define the objectives to schedule our test activities and understand the goals of the customer and the project.

The tasks to perform during the planning include:

- Identify the objective and goal of testing.
  - **What is the goal for the regression testing?**
    The goal for this regression testing is to validate Computers Database Page works correctly and that it is possible to perform the different CRUD actions through the UI.

  - **What kind of coverage does the software team want for its regression testing?**
    Cover all scenarios that are required to perform CRUD actions.

- Schedule QA related activities.
  - **How the plan would be maintained?**
    We must keep the regression scheduled and established according to QA activities. The frequency of regression testing is related to the frequency with which modifications or updates are made to the system.
    Depending on the changes that are made, only the module that is affected would be executed. And a test scenario for each of the other modules would be added to the execution, to verify that they are not affected and to optimize the execution time.

  - **How would be integrated with deployment pipeline?**
    Se debe de integrar en el release pipeline justo después de hacer deploy a los ambientes de desarrollo, qa, uat y staging para así adelantar el proceso de testing y verificar de una vez si algo hizo falta en el release plan.

Implementation

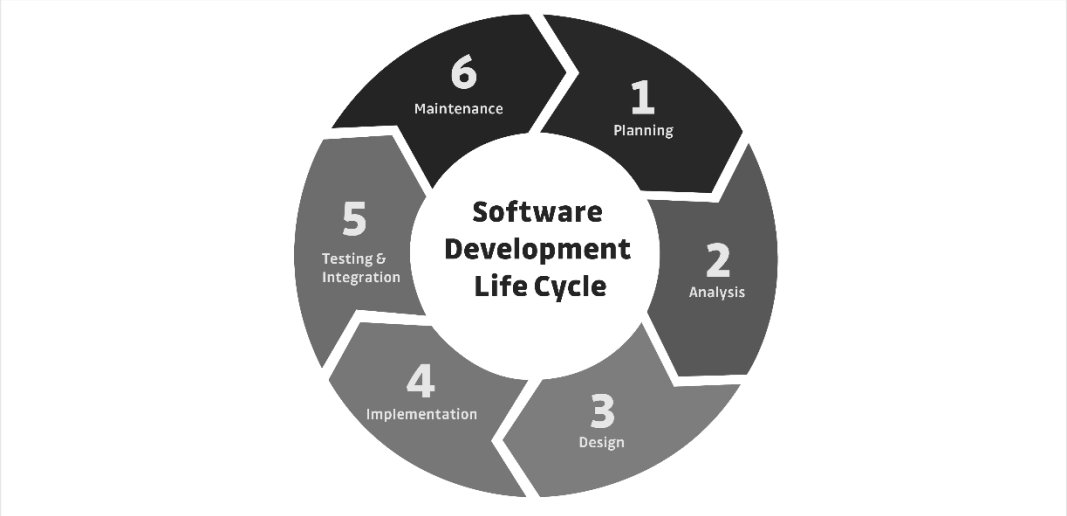
Implement the test suite and organize sections for the test cases in order to efficient test execution. In this case, the test suite will be a logical collection of test cases which naturally work together. We expect that test suites share data and a common high-level set of objectives.

When implementing a sorted test suite, such as the one shown in the previous section, we can

- Keep track of the associated data for a given set of tests
- Provide traceability of tests, test results and defects

Examples of how the QA team and the suite of regression tests produced by the enactment of this plan fit into the overarching SDLC

Considering the diagram shown in the following image, where the stages of the software development life cycle are shown, the following examples can be given.



1. When the testing of step 5 fails, it is required to return to step 4 to perform a new implementation which must be tested later, it is here where it would be useful to run the regression test plan because in some way, we would have more coverage to determine that other areas have not been affected after new implementations.

2. When making an implementation that arises from a bug, you can run it to consider the scope of the bug and that other modules may be affected by this failure.

Execution

For the execution we should have in mind to develop and prioritize our test cases, this refers to the following.

- Execute the test suites and individual test cases, following test procedures. Do not forget the priority.
- Log the outcome of test execution and record the versions of the software under test, test tools and test ware.
- Compare actual results, what happened when we ran the tests, with expected results, what we anticipated would happen.
- If there are differences between actual and expected results, report discrepancies as bugs. We analyze them to gather further details about the defect, reporting additional information on the problem, identify the causes of the defect.
- We need to re-execute tests that previously failed to confirm a fix. We need to re-execute corrected tests and suites if there were defects before.

We'll also set up a test execution schedule, to organize execution and to estimate points for each test case.

Automation

Regression testing is often a lengthy process and incorporating test automation has several advantages. Firstly, the execution capacity increases, while the duration of the tests is reduced. Tests can be run as many times as required without putting wear and tear on the equipment.

We will work with Cypress, a Javascript end-to-end testing tool. In other words, it allows to check that the performance of a newly developed software product is good and corresponds to the initial requirements.

Through the integration of Cucumber, it will make it easier for QA to write automated test cases. Cucumber is a testing approach/tool that supports Behavior Driven Development (BDD). It provides a way to write tests that anybody can understand, regardless of their extent of technical knowledge. It runs automated acceptance tests written in BDD format.

The automated features for the mentioned test scenarios can be found in the project under the following path “Automation Test\cypress\e2e\features\regression tests”.