TST | Xper IM | Checklist for reviewing automated tests

Checklist version below in Excel format:

Checklist - Automation Review.xlsx

Use this checklist whenever starting an automated test review:

Header

- Verify if the header follow this example:

  // Script creation: <fill with creation date>

  // Last maintenance: <update last maintenance date>

- Check if the "Main Objective" is the same as the one in Azure DevOps.
- Check if the "USEUNIT" is correct (*use the CommonUseunit that contains all use units).*

Global Variables

- Check if all declared variables are used in more than one Step.
- Check if the name of the variables makes sense and if they are declared with the first lowercase letter.
- Check if the variables are receiving the value at the time they will be used. (Because if a value is received during the declaration it can cause problems in the execution).

unitCreate

- If it exists, check if it is used only to assign values to global variables.

preConditions

- Check if all pre-conditions of the Test Case are being made in the preCondition function and if no preconditions are being made without being in the TC.
- Check if the error messages are declared correctly and if they are being specific. Examples:
  - *Good Example: addErrorDef ('Precondition failed!', 'Was not possible to open the "Inventory" tab.');*
  - *Bad Example: addErrorDef ('Precondition', 'Screen not opened.');*

postCondition

- Verify that all system configuration changes are being undone.
  - *Example: In the preCondition, the system language was changed to German. In postCondition then the language must be changed to English again if this has not been done in any step of the TestCase.*

- If there is an error message, check if it is declared correctly and if it is being specific.

Note: The postCondition is not required to be mentioned in the TC and it must not be a Step.

"main" function

- Check if the "unitCreate" function is being called if it exists.
- Check that the text in the "reportHeader" is the same as in Azure DevOps (including line breaks).
- Check if the TestCase parameter of the reportHeader is being filled in the template pattern.
- Check if the "preCondition" function is being called if it exists.
- Check if all test case steps are being called.
- Check if the "postCondition" function is being called if it exists.

Steps

- Verify that each step does exactly what is described in the Azure DevOps step.
- Check if there are validations for the actions:
    - *Example: TestCase asks to create and start the case but in automation, it was checked if the case was started before going to the next action.*
- Check if the error messages are declared correctly and if they are being specific in case an error occurs anyone can look at the log and identify what is wrong and mainly to follow the ***guidance 0730-02-G2 Good Documentation Practices****.*
    - *Bad Example: addErrorDef ('Step 7 failed!', 'The "Barcode Single Scan" window was not displaying the correct values!');*
    - *Good Example: addErrorDef ('Step 7 failed!', 'The "Barcode Single Scan" window was not displaying the correct values! \ N' + Expected GTIN: 12445678901234, ExpDate: 500120, Serial: Ser12345. \ n '+ 'Found: GTIN 123456, ExpDate: 010120, Serial: Ser123');*
- Check if the Expected Result is the same as it is in Azure DevOps (including line breaks).
- Check if Actual Result is the same text as Expected Result but in the past tense (including line breaks). NOTE: Actual may have extra information.
- Check if the declared variables (if any) are being used.
- Check if the Expected Result (evidence) is really efficient.
    - *Bad example: "Xims log was found".*
    - *Good example: "Xims log was found - create date: 11/13/2019".*
- Check if the last step has the "final" parameter at the checkpoint.

Other checks

- Check if the name of new functions starts with a lowercase letter.
- Check if the component mapping follows the pattern of those that already exist:
    - function name starting with a lowercase letter;
    - mapping always using the "parent" component plus the name of the mapped component;
    - mapping always using simple names to identify the component's screen if it is a more generic component.
        - *Example: function btnOkInventoryScreen ()*
- Check if the new test has been included in a test suite.
- Check if the "Script.tcScript" file was committed together when a new test was created.
- Check if the file paths in script.tcScript are in the default.
- Check if the file "CommomUseUnit" was changed if a new screen mapping or a new generic function file was created.
- If there is component mapping, check if that component is not already mapped in another method.
- Functions and mappings cannot work expecting fixed screen or record positions. example:
    - When using the record, use a fixed position of a component on the screen.
    - When you always use the registration of the first position of the research believing that it will always be the same, instead of searching for what you really want.

- Check if a code structure always repeats (Ctrl + C + V). Redundancy.

- Avoid unnecessary "Delays".
- When automating tests in Hemoconditions screen always use low level procedures for executing actions like clicking or changing grid tabs.
  This is necessary because this screen can be updated with information coming from Hemo and this can cause the focused component to change and the test execution can fail.