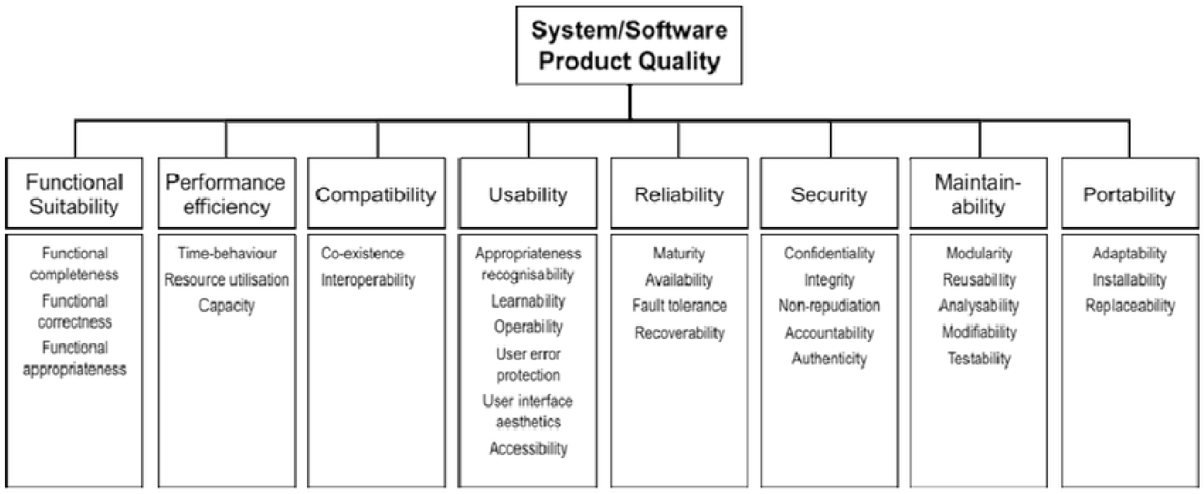
Intro

If we want to be sure that the test coverage of the functionality is enough we need to follow the QA standards.
For example the one of the most famous SO/IEC 25010.

Decomposition of this feature for the different levels of the testing will be below.



Performance efficiency
Compatibility
Functional suitability
Usability
Reliability
Security
Maintainability
Portability

Each of these levels is responsible for the area of the requirements. Not all of them need to be tested, it depends on the context.

Documentation review & questions

- Define how often the new configurations will be update, depends on it we need to make our test framework flexible
- What is the expected result if we will not provide a configuration for the rules or products?
- What will happen if the configuration of those files will be broken? Syntax or smth else

- Define a scheme of the Yaml to use it as a test data source for our tests
- Is there only a fund as a currency?
- This feature describes only an addition of the product but there are also other operations connected with it. Deletion, Submission, Cancelation, etc
- Define a contract of the interface, what kind of the parameters this function has
- Define an integration of the modules and applications, to find the best scenarios for integration test
- Define how this configuration will deploy in production to make an integration tests for configuration delivery scenarios
- The values in the table are the values without discount?
- What is the maximum quantity of items to be submitted?
- How does the system react to the invalid id as an item for addition?
- How the system works with users, is it session based or something else? How do they interact with this function in the whole business scenario ?
- Does this feature work if we will update the file in real time?
- If the system uses a configuration, what will happen if we provide a product which does not exist?

Unit tests

[Test Data]
Coffee - free rule product
Green tea - reduced price rule product
Strawberries - fraction price rule

Scenario: Add 1 product without any discount configuration
Expected result:Cart contains 1 item, total price = item price

Scenario: Add 1 product with free rule discount
Expected result: Cart contains 2 items, total price = item price

Scenario: Add N items for ReducedPriceRule configuration
Expected result: Cart contains N items, total price = item price * N

Scenario: Add N+1 items for ReducedPriceRule configuration
Expected result: Cart contains N items, total price = changed price from configuration * N

Scenario: Add N items for FractionPriceRule configuration
Expected result: Cart contains N items, total price = item price * N

Scenario: Add N+1 items for FractionPriceRule configuration
Expected result: Cart contains N items, total price = % of item price from configuration * N

Scenario: Add empty item as a value depends on the interface of the function
Expected result: Cart contains the number of items was before the addition

Scenario: Add N+1 items for FractionPriceRule + 1 product without discount + 1 product with free rule + N+1 items for ReducedPriceRule
Expected result: Cart contains N * 2 + 4 items, N+1 - green teas , N+1 - strawberries, 1- coffee, 1 product without discount, total price changed value for green tea * (N +1) + (% of strawberries from configuration * (N +1) + 3.11 + price for the product without discount

Scenario: Add N+1 items for FractionPriceRule configuration and delete 1 item
Expected result: Cart contains N items, total price = item price * N

Scenario: Add N+1 items for ReducedPriceRule configuration and delete 1 item
Expected result: Cart contains N items, total price = item price * N

Scenario: Add 1 item for the product with free rule discount configuration and delete 1 item
Expected result: Cart contains 0 items, total price = 0

Integration tests

Provided as an example in the code repo
+
Checking of the integration file system with the configurations. Depends on the specific of the file loading
+
Docker containers for the modules & applications wich works closely in the business scenario

Performance tests

Scenario: With the help of some some tools make 1 bln requests to the function to determine how the system will work with this type of configuration (file based)
Expected result: No errors, time of execution bellow then the benchmark

Scenario: Depends on the how this function works with external modules/application make a stress test of UI
Expected result: connection between modules/application does not broken

Scenario: Make a throttling between file system and the functionality and make a 1000 requests for adding items to the cart
Expected result: Delaying of the file reading does not affect end user

Availability tests

Scenario: Make a 1 bln requests with add/delete items in the cart under 1 user
Expected: The resulted cart not broken, system also not broken

Scenario: Make a 1 bln requests with add/delete items in the cart under 1000 users
Expected: The resulted cart not broken, system also not broken