TEST WORKFLOW

SQS TestWORKFLOW is a tool designed to support the entire software validation process, assisting the user in automated testing and the management of requirements, tests and documentation.

This tool integrates requirements definition and management, the organisation of the testing process, the automated execution of test programmes, the analysis of results, and the generation of multiple reports.

Traceability is provided throughout the process, on account of the recorded relationships between test cases, requirements and executions.

SQS TestWORKFLOW's intuitive interface enables every kind of user to understand the process and become familiar with the tool without being an IT specialist, while the systematic methodology which the tool implements accomplishes a good structured test process.

Requirements definition and management

Good requirements management is one of the keys for a successful validation process.

Requirements express the requested behaviour of the system. Every requirement models a function of the system, so that the complete set of requirements should completely cover the whole functionality of the system.

In the validation of a system one must be sure that every requirement is tested; this can be checked through the relation between requirements and test cases.

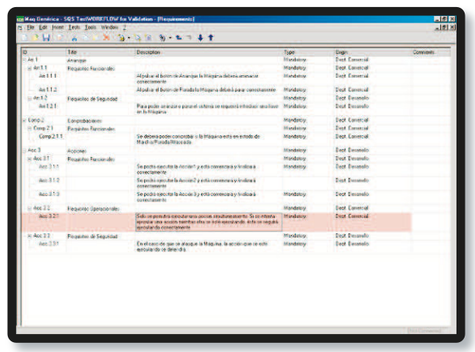
In **SQS TestWORKFLOW**, requirements are defined using a spreadsheet-like template which contains all relevant information and enables them to be easily administrated: requirements can be grouped together to show the similitude or dependence between them, and they can be moved by dragging and dropping.

Changes or new requirements are also very easy to implement.

Tests Organisation

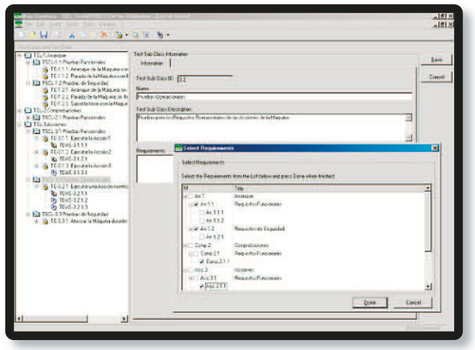
Test structures are based on a tree hierarchy, containing the following items: Test Class, Test Sub Class, Test Case and Test Data.

Test Class: A Test Class is the highest level of a test structure. It contains one or more Test Sub Classes. A Test Class represents a logical group of system requirements.



Test Sub Class: A Test Sub Class is a subset of a Test Class. It's used to refine the division of the requirements into smaller sets because, for a large system, the number of requirements is too high and the division in Test Classes not accurate enough.

Test Case: A Test Case serves to prove specific system functionality. Every requirement of the system must be contained in at least one Test Case, though each Test Case may cover more than one requirement.



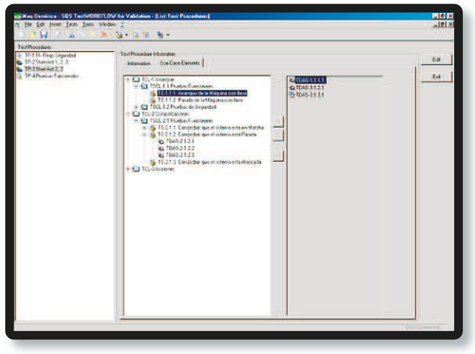
Test Data: Every Test Case, which is a logical definition of a requirement, contains a set of Test Data, executable tests with specific values that depend strongly on the system implementation. A set of Test Data must cover every possible combination from the Test Case it belongs to.

The executable elements of Test Data are called **steps**. Every step performs an action in the system, like for example clicking on a button or making an input. In SQS TestWORKFLOW there are initialisation steps and execution steps.

Assuming an initial state of the system, which if necessary, can be reached by executing the initialisation Steps, Test Data performs execution steps to bring the system into its final state.

Steps can also be grouped in Macros. A macro is a sequence of Steps that need to be executed very frequently. A typical use of a macro can be, for example, part of an initialisation, which is to be executed identically for most of the Test Data.

Test Data can be grouped in series and be saved into what is known as **Test Procedure**.



There are 2 types of Test Procedures: Collections and Use Cases.

Collections: consist of a series of Test Data, which are independent from one another.

They can be **Test Collections**, whereby the Test Data is selected from a test-oriented view, or **Requirements Collections**, whereby the Test Data is selected from a requirement-oriented view.

A **Use Case** is a series of Test Data that simulates the behaviour of a user. These Test Data execute the actions that the user performs on the system one after the other in a specific order.

Execution

Once the Test Data have been defined, a test execution may be launched.

Tests can be executed automatically, but the user can also control the execution manually (semi-automatically).

In both cases, **SQS TestWORKFLOW** creates a log file containing the test results.

For an execution, a single piece of Test Data can be selected, but usually an execution contains several pieces of Test Data, selected to test a group of Test Cases or Requirements.

In SQS TestWORKFLOW there are the following types of executions:

- Execute by Tests
- Execute by Test Procedures
- Execute by Requirements

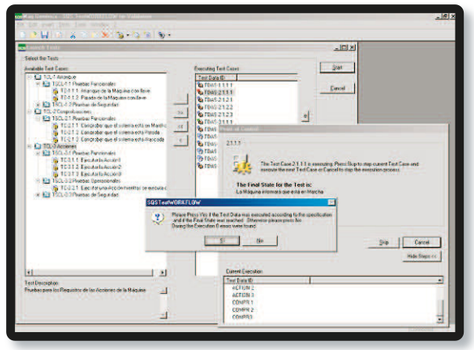
Analysis of results

SQS TestWORKFLOW also evaluates the test results.

In an automated execution, the tool performs the comparison between the expected and the generated results and determines whether the Test Data has passed or failed.

For a semi-automated execution, the tool shows the generated results of Test Data and asks the user to state whether they match the expected results. The user then decides whether each Test Data has passed or failed.

The complete set of tests is than evaluated as passed or failed according to the system acceptance criteria.

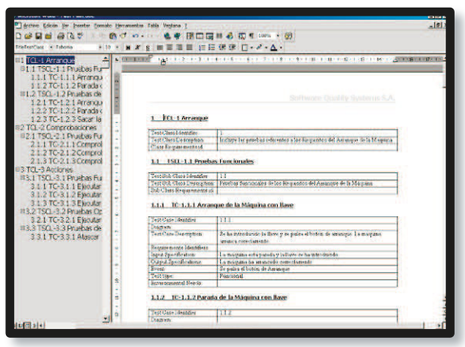


Reports

The user can always get information about the current status of the test organisation and execution through several reports.

These reports provide thorough information about the system, and facilitate traceability.

Reports are exported to Microsoft Word, thus providing a simple and standard document, which every user can work with. Some typical reports are:



Test Design Specification, detailing the test structure of Test Classes, Test Sub Classes and Test Cases.

Requirements Specification Report, documents all system requirements and their relationships.

Executed Tests Report, lists all test items and shows the execution status of all Test Data. It also provides statistics about the system, including the quantity of Test Data that have states passed, failed or not executed.

Further reports can also be defined and customised by the user.

Glossary

Validation: It is the process of testing a system to ensure that the system meets the requirements which define its behaviour.

Test Class: A Test Class is the highest level of a test structure. It contains one or more Test Sub Classes. A Test Class represents a logical group of system requirements.

Test Sub Class: A Test Sub Class is a subset of a Test Class. It's used to refine the division of the requirements into smaller sets because, for a large system, the number of requirements is too high and the division in Test Classes not accurate enough.

Test Case: A Test Case serves to prove specific system functionality. Every requirement of the system must be contained in at least one Test Case, though each Test Case may cover more than one requirement.

Test Data: Every Test Case, which is a logical definition of a requirement, contains a set of Test Data, executable tests with specific values that depend strongly on the system implementation. A set of Test Data must cover every possible combination from the Test Case to which it belongs.

Requirement: Requirements express the required behaviour of the system. Every requirement models a function of the system, so that the complete set of requirements should completely cover the entire functionality of the system. In the validation of a system, one must make sure that every requirement is tested; this can be checked through the relation between requirements and test cases.

Test Procedure: Series of Test Data which will be saved. There are 2 types of Test Procedures: Collections and Use Cases.

Use Case: Series of Test Data that simulate the behaviour of a user.

Report: Document containing information about the current status of the test organisation and execution. Reports provide thorough information about the system, and facilitate traceability.

Environment: All test structures, requirements, reports and other information related to a project, where a project is a clearly defined set of software programmes and their requirements and/or specifications for a certain version/release, is contained in a file named Environment.

An Environment must be defined for each project.

Spain
Av. Zugazarte 8 - 1º, 48930 Getxo, Vizcaya
T +34 94 480 46 17

Spain
Alcalá 35, 28010 Madrid
T +34 91 531 05 12

Belgium
Boulevard de la Cambre 52, 1000 Brussels
T +322 645 08 71

Germany
Rosenheimer Str. 145 C, 81671 München
T +49 (0) 8923067842