Software Qualification Test Strategy (SWQTS)

|  |  |  |
| --- | --- | --- |
| **OP’nSoft Project information** | | |
| **Project ID** | **Project Name** | **Project Manager** |
| [Type Project ID] | [Type Project Name] | [Type Project Manager] |
| **Field of application:** [Type field of application] | | |
| **Customer Name** | **Project Start Date** | **OP’nSoft project ID** |
| [Type Customer Name ] | [Type Project Start Date ] | [Type OP’nSoft project ID] |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Function | Signature |
| Edited by |  |  |  |
| Reviewed by |  |  |  |
| Reviewed by |  |  |  |
| Reviewed by |  |  |  |
| Approved by |  |  |  |

# Template Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Version | Author | Section | Description / Task ID |
| 27/11/2023 | 1.0 | F. Glénard | All | Template definition for OP’nSoft |
|  |  |  |  |  |

# Document Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date  **(yyyy-dd-mm)** | Version  (x.y) | Status | Author | Section | Description / Task ID |
|  |  | Choose an item. |  |  |  |
|  |  | Choose an item. |  |  |  |
|  |  | Choose an item. |  |  |  |
|  |  | Choose an item. |  |  |  |

# Table of Contents

[Template Revision History 2](#_Toc151995290)

[Document Revision History 2](#_Toc151995291)

[Table of Contents 3](#_Toc151995292)

[1 Introduction 4](#_Toc151995293)

[1.1 Purpose and Scope 4](#_Toc151995294)

[1.2 Review and Approval 4](#_Toc151995295)

[1.3 Referenced documents 4](#_Toc151995296)

[1.4 Applicable documents 5](#_Toc151995297)

[1.5 Abbreviations 5](#_Toc151995298)

[2 Qualification test strategy 6](#_Toc151995299)

[2.1 Test objective 6](#_Toc151995300)

[2.2 Test Prioritization 6](#_Toc151995301)

[2.3 Variant Management 6](#_Toc151995302)

[2.4 Pre-conditions for qualification testing 6](#_Toc151995303)

[2.5 Test scope 7](#_Toc151995304)

[2.6 Test environment and tools 7](#_Toc151995305)

[2.6.1 Environment 7](#_Toc151995306)

[2.6.2 Tools 7](#_Toc151995307)

[2.7 Test Design Strategy 8](#_Toc151995308)

[2.7.1 Test case design method activities 8](#_Toc151995309)

[2.7.2 Test case execution method categories 9](#_Toc151995310)

[2.7.3 Test cases content 9](#_Toc151995311)

[2.7.4 Test case priority 10](#_Toc151995312)

[2.7.5 Test criteria 10](#_Toc151995313)

[2.8 Test case/step result defect severity 10](#_Toc151995314)

[2.9 Regression Test Strategy 11](#_Toc151995315)

# Introduction

## Purpose and Scope

**Purpose:** The purpose of this Software Qualification Test Strategy is to define the overall approach that will be taken by the Test Team when defining the qualification testing of the [Type Project Name] project. The Software Qualification Test Strategy is created during the planning phase of the project.

Software qualification test is the verification of the correct implementation of software requirements including the functional safety requirements.

**Scope:** This Software Qualification Test Strategy is valid for the [Type Project Name] project, it affects the software test activities and is part of the OP’nSoft test process.

## Review and Approval

The Software Qualification Test Strategy is intended for the project team and should be modified/adapted according to the project organization needs.

After the initial creation or any update afterwards, the document needs to be approved by all the following parties below:

* Software Proxy Product Owner
* Software Product Owner
* Software Quality Engineer

## Referenced documents

<In the table below, list the all the technical documents, norms, standards, etc., that were used to define this document>

|  |  |  |  |
| --- | --- | --- | --- |
| **Document name** | **Version** | **Date** | **Storage path and link** |
| Automotive SPICE - Process Reference and Assessment Model | 3.1 | 01/11/2017 | [Link](https://confluence.engine.capgemini.com/x/oFsoAQ) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Applicable documents

<In the table below, list the all the project specific documents that are cited in this document or that were used to define this document>

|  |  |  |  |
| --- | --- | --- | --- |
| **Document name** | **Version** | **Date** | **Storage path and link** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Abbreviations

<In the table below, list and describe the abbreviations that are used in the document>

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| ASIL | Automotive Safety Integrity Levels |
| FuSa | Functional Safety |
| HIL | Hardware In the Loop |
| MIL | Model In the Loop |
| SIL | Software In the Loop |

# Qualification test strategy

## Test objective

|  |  |  |  |
| --- | --- | --- | --- |
| **Linked process** | **Objective** | **Measurable target** | **Responsibility** |
| SWE.6  Software Qualification Test | Coverage of all software requirements and functionality | 100% of software requirements shall link to test cases |  |

## Test Prioritization

The priority of specifying test cases, test case implementation and test execution will be decided by the Software Validation Engineer and Software Product Owner.

The following points shall be considered while deciding priority.

1. Requirements are assigned to specific releases, based on this assignment test cases are created & tested to the respective releases

2. Tests related to safety requirements shall be treated with the highest priority.

4. Priority of re-tests: severity/priority attribute of tasks/defects may indicate which tests to be adjusted and executed first.

5. New tests may be more important than re-tests.

## Variant Management

Variants are listed below for [Type Project Name]

<List of Variant>

## Pre-conditions for qualification testing

Before starting qualification testing, the following conditions have to be met:

1. The software requirement specification is baselined.
2. The software integration tests are passed, and test results are baselined.

## Test scope

The software qualification test is mainly a black box test. The test object is regarded as a black box. Regardless of the internal logical structure of the software, only the requirements specifications are used to check whether the function of the test object meets its functional description.

<Fill the following table with the main functions of the software, their key test points, test environment and test mode. The first 2 lines show an example that you should remove when filling the template>

|  |  |  |  |
| --- | --- | --- | --- |
| **Software Main Functions** | **Key Test Points** | **Test environment** | **Test mode** |
| Battery Data Collection | Cell Voltage Sampling: Validity of collected data (including the validity of the first frame of data, the sending cycle, and the sampling range) | HIL | Manual test |
| Model Temperature Sampling: Validity of collected data (including the validity of the first frame of data, the sending cycle, and the sampling range) | HIL | Manual test |
|  |  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Test environment and tools

### Environment

<Describe/illustrate the test environment that will be used for qualification testing>

### Tools

<Fill the table below with information about the tools that will be used for qualification testing>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Device** | **Internally produced / Manufacturer** | **Specification / Property / Version** | **Description of the tool application** | **Availability** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Test Design Strategy

The purpose of the software qualification test is to ensure that the software is tested to provide evidence for compliance with the software requirements including functional safety requirements.

|  |  |
| --- | --- |
| **Test level** | Software Qualification test |
| **Test object** | Software function |
| **Test basis** | Software Requirements Specification including functional safety requirements |
| **Test Objective** | Carry out a series of tests on the whole software to assure the coverage of all software requirements and functionality for the given release |
| **Test method context** | Black box testing |

### Test case design method activities

**For non-functional safety related test** at the software qualification test level, test cases can be derived using an appropriate combination of methods, as listed in the table below :

|  |  |
| --- | --- |
| **Test Design Method activities** | **Description** |
| Equivalence class generation and analysis | Divide a large dataset into different ranges where the testing effects are similar, making each range a representative group. Then, pick specific data points from each range to reflect the test results within that range. |
| Analysis of boundary values | The boundary value analysis method is a black box test method for testing the input or output boundary value. Check whether the upper, lower, and off points of the data can be processed correctly. |
| Causality graph method | Use a graphical method to illustrate different input combinations and then create a decision table to design the corresponding test cases. |
| Error guessing based on knowledge or experience | During program testing, individuals can use their experience and knowledge to anticipate potential errors in the program and develop specific test methods to check for these errors. |

**For functional safety testing** at the software qualification test level, test cases can be derived using an appropriate combination of methods, which listed in the table15 Methods for deriving test cases for integration testing of ISO26262-6:2018(E) should be considered.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Design Method activities** | **Description** | **ASIL** | | | |
| **A** | **B** | **C** | **D** |
| Analysis of requirements | Analyze the software requirements, functional safety to develop relevant test case. | x | x | x | x |
| Generation and analysis of equivalence class | Divide a large dataset into different ranges where the testing effects are similar, making each range a representative group. Then, pick specific data points from each range to reflect the test results within that range. |  | x | x | x |
| Analysis of boundary values | The boundary value analysis method is a black box test method for testing the input or output boundary value. Check whether the upper, lower and off points of the data can be processed correctly |  |  | x | x |
| Error guessing based on knowledge or experience | During program testing, individuals can use their experience and knowledge to anticipate potential errors in the program and develop specific test methods to check for these errors. |  |  | x | x |
| Analysis of functional dependencies | Whether the software functions meet customer requirements, new and changed functions should be covered |  |  | x | x |
| Analysis of operational use cases | Analyze the operation process and complete the use case design |  | x | x | x |

### Test case execution method categories

|  |  |  |
| --- | --- | --- |
| **Execution Method** | **Description** | **Relevant for the project** |
| Requirement-based test | Software requirements and specific test requirements are used for testing. This includes testing related to cybersecurity and functional safety requirements. When it comes to cybersecurity requirements, test cases are designed based on attack patterns or STRIDE threat modeling. Additionally, cybersecurity test cases should also be created to verify the effectiveness of cybersecurity controls or measures | <Yes/no + justification if no> |
| Fuzz Test | Fuzz testing, or fuzzing, is a method in software testing where you intentionally feed incorrect or random data (called "fuzz") into a software system. The goal is to **find coding errors and security vulnerabilities**. In fuzz testing, the emphasis is primarily on the data that comes from external sources or interfaces of the software. You provide a wide range of valid and invalid input data, and you also randomize the timing of signals. Then, you check how the software behaves under these different conditions to identify potential issues. | <Yes/no + justification if no> |
| Penetration Test | Penetration testing assesses the security of an application to uncover vulnerabilities. It's crucial to confirm the **software's resistance to cyberattacks** before system-level testing. This involves verifying essential cybersecurity functions like key management, secure configuration, secure boot, software updates, secure communication channels, memory access rights, diagnostic services, and protection against brute force attacks. | <Yes/no + justification if no> |
| Fault injection test | Tests related to software injection into hardware or other interface faults (including the functional safety fault） | <Yes/no + justification if no> |

### Test cases content

The test cases specification must contain the following information:

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **ID** | Unique test case ID |
| **Title** | Title of the test case |
| **Status** | Indicate if the test case is Active / Not Active |
| **Level** | **Qualification test level** |
| **Purpose** | Describe the purpose of the test case |
| **Priority** | Indicate if the test case if of High, Medium, or Low priority for it’s execution |
| **Environment** | Indicate the test environment for the test case: HIL / MIL / SIL / vehicle testing… (refer to sections 2.6.1 & 2.6.2) |
| **Method** | Describe the method that will be used for conducting the test case (see section 2.7.2) |
| **Pre-condition** | Define the condition(s) that have to be met to start the test |
| **Functional module** | Indicate the functional module to which the test case belongs |
| **FuSa** | Indicate if the test case is Functional Safety relevant |
| **Cybersecurity** | Indicate if the test case is Cybersecurity relevant |
| **Steps** | Create as many steps as required |
| **Steps titles** | Title of each step of the test case |
| **Steps descriptions** | Describe each step of the test case |
| **Steps expected results** | Describe the expected result of each step of the test case |

### Test case priority

|  |  |
| --- | --- |
| **Test priority** | **Description** |
| **High** | Basic functions of the software, that affects the entire software function or cybersecurity and functional safety function |
| **Medium** | Basic functions of the module, tests that affect the module function. |
| **Low** | Small area of influence, could continue to perform tests on other functional modules |

### Test criteria

|  |  |
| --- | --- |
| **Test completion criteria** | There are a set of releases in the project. For each release a specific set of software requirements are delivered. The test completion applies the test objectives in section 2.7 only for the elements that need to be covered for this specific release. |
| **Test start criteria** | Complete module integrated software shall be available without any error and warnings, and source code should compile with test environment |
| **Test end criteria** | All test objectives for software qualification testing are achieved according to the section 2.7 test objectives. In addition, all bugs should be closed and all expected and actual results are captured and documented with the test script, 100% of software requirements shall link to test cases |
| **Test abort criteria** | if any stopper bug/blocker arises, testing shall be aborted |
| **Test restart criteria** | if any stopper bug/blocker arises, testing shall be aborted and after fixing bug, testing shall be restarted |

## Test case/step result defect severity

For detail about software defect tracking, please refer to the [SUP.9 – Problem Resolution Management Process](https://confluence.engine.capgemini.com/x/vKtVAQ).

|  |  |
| --- | --- |
| **Defect severity** | **Description** |
| **Blocker** | Critical defects causing system or application crashes, freezes, system hangs, or loss of data, complete loss of major functions, cybersecurity function or functional safety abnormal |
| **High** | A major defect refers to the failure to implement a function or feature, the loss of a major function, causing a serious problem. |
| **Normal** | Less serious errors. Although such defects do not affect the basic use of the system, but if not better. |
| **Minor** | Some minor problems have little effect on the function, and the product and attributes can still be used, such as individual typos and irregularly arranged text. |

## Regression Test Strategy

Regression test shall verify that implemented SW changes have no unintentional "side effects" on software function that have not been changed.

A regression test shall be performed on basic functionality and safety relevant implementations.

Test cases, relevant for regression test, shall be marked in the test specification.

During regression testing with unchanged test cases shall prove as per below points.

1) Unchanged test cases running on unchanged code shall "PASS"

2) Unchanged test cases running on changed code shall "FAIL"

3) Changed test cases on changed code shall "PASS"

The Regression testing shall include re-running of previously completed tests and to verify fixture of current reported defects and re-emergence of previously fixed defects. Regression test shall always be performed when any change is done in Software requirement for respective modules in every milestone of the project.