SYSTEM TEST PLAN

ROBUST SOFTWARE PROJECT

DATE: 2020-11-26
DOC. REF.: DOC_REFERENCE_NBR
STATUS: APPROVED
PAGES: 18
INFORMATION CLASSIFICATION:CONFIDENTIAL
VERSION: 1.0

APPROVAL

VERSION	NAME	FUNCTION	SIGNATURE	DATE
1.0		Project Manager		

AUTHORS AND CONTRIBUTORS

NAME	DESCRIPTION	DATE
Nuno Silva	Author	
	Reviewer and Contributor	
	Reviewer	
	Reviewer	
	Reviewer	

COPYRIGHT

REVISION HISTORY

VERSION	DATE	DESCRIPTION	AUTHOR
0.1	20/01/2020	First draft	
0.2			
0.3			

TBC/TBD

ID	SECTION/TABLE/FIGURE	TBC/TBD	REMARKS
TBC_001	TABLE 5	TBC	The list of requirements that are not testable might not be closed and need further review.
TBC_002	TABLE 6	TBC	The list of requirements that are partially covered might not be closed and need further review.

TABLE OF CONTENTS

TABLE OF TABLES

TABLE OF FIGURES

1. INTRODUCTION

1.1. OBJECTIVE

This document primarily presents the overall test plan for PROJECT_SR. The objective of the above-mentioned task is to completely define an overall test approach, identify the features to be tested, outline test success/failure criteria to be employed, detail the test deliverables and tasks, present and detail the test environment and its needs, provide a schedule of the testing activities and risks associated with testing activities.

Based on all the above, the general objective of this document is to define a project-wide test plan explaining, in detail, the approach for the testing activities. This document serves as input for the design of the System and Software Test Specifications as well as for the test execution and Test Report elaboration.

1.2. SCOPE

This document lies within the scope of the validation testing activities according to the railway CENELEC standards for the PROJECT_SR. The testing activities are split into different work packages which differ in the planned timeframe as well as in the approach followed, while also covering different test objectives. This is thoroughly detailed in the test plan presented in this document.

1.3. AUDIENCE

The intended audience of this document is:

- Project team;
- Customer Team;
- Others.

1.4. DEFINITIONS AND ACRONYMS

TABLE 1 presents the list of definitions used throughout this document.

NAME	DESCRIPTION
Applicable Document	A document is considered applicable if it complements this document. All its content is directly applied as if it was stated as an annex of this document.
Reference Document	A document is considered a reference if it is referred but not applicable to this document. Reference documents are mainly used to provide further reading.

TABLE 1: Definitions

TABLE 2 presents the list of acronyms used throughout this document.

ACRONYM	DESCRIPTION
AD	Applicable Document
API	Application Programming Interface

ACRONYM	DESCRIPTION
CENELEC	European Committee for Electrotechnical Standardization
CRC	Cyclic Redundancy Code
HW	Hardware
JDB	Java Debugger
JMS	Java Messaging System
JRE	Java Runtime Environment
OS	Operating System
PID	Process ID
RD	Reference Document
SAS	Software Architecture Specification
SIL	Safety Integrity Level
SPAE	SW Product Assurance Engineer
SSL	Secure Sockets Layer
SW	Software
SwRS	Software Requirements Specification
SwRTS	Software Requirements Test Specification
SyRS	System Requirements Specification
SyRTS	System Requirements Test Specification
TBC	To be confirmed
TBD	To be defined
THR	Tolerable Hazard Rate
TLS	Transport Layer Security
V&V	Verification & Validation

TABLE 2: Acronyms

1.5. DOCUMENT STRUCTURE

Section 1 (Introduction) presents this document.

Section 2 (Test Plan) outlines the test plan and strategy.

1.6. APPLICABLE DOCUMENTS

TABLE 3 presents the list of the documents that are applicable to this document. A document is considered applicable if it complements this document. All its content is directly applied as if it was stated as an annex of this document.

Note: The version of some the applicable documents is the one at the data of approval this document, as captured in the Configuration Items List [RD-2], thus, specific versions are not indicated in this table.

APPLICABLE DOCUMENT	DOCUMENT NUMBER
[AD-1] TBC	DOC_REFERENCE_NBR
[AD-2] TBC	DOC_REFERENCE_NBR

TABLE 3: Applicable Documents

1.7. REFERENCE DOCUMENTS

TABLE 4 presents the list of reference documents. A document is considered a reference if it is referred but not applicable to this document. Reference documents are mainly used to provide further reading.

Note: The version of some the reference documents is the one at the data of approval this document, as captured in the Configuration Items List [RD-2], thus, specific versions are not indicated in this table.

REFERENCE DOCUMENT	DOCUMENT NUMBER
[RD-1] Railway application - Communications, signalling and processing systems- Software for railway control and protection systems	EN50128:2011
[RD-2] PROJECT_SR Configuration Items List (CIL)	DOC_REFERENCE_NBR

TABLE 4: Reference documents

2. TEST PLAN

The proposed Validation Testing activities can be seen as divided into different sub-work packages or phases, as shown in FIGURE 1.

Prior to the beginning of this work package, unit/module tests must be executed. These unit/module tests are not depicted in the below figure because they belong to the previous Implementation work package. However, they are mentioned in this document because it is a relevant testing phase.

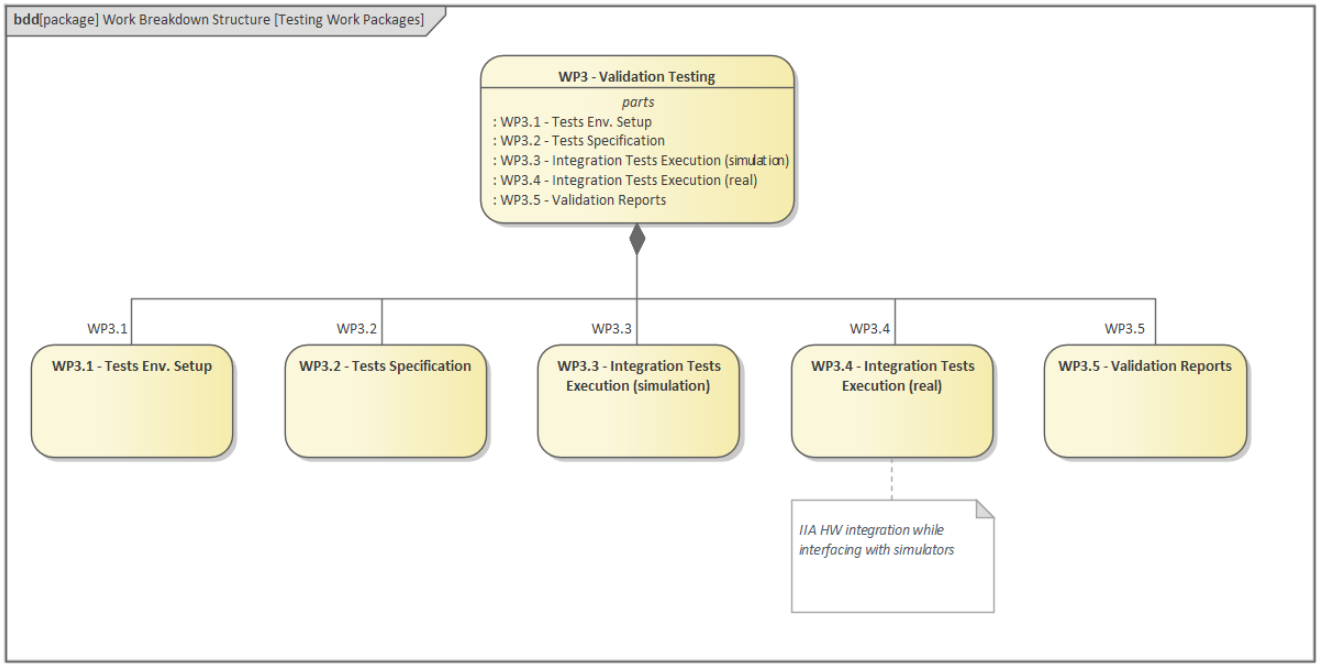

FIGURE 1: Testing Work Packages

In this case, the underlying plan described throughout this document applies to every test activity, ranging from Module to Software and System level specification and testing. Since there are different levels of testing, a clear distinction between them is made, focusing on the purpose and scope of System and SW testing, in the context of this project. This will translate into a common understanding of the content of the deliverable documents and will also explain the mapping relationship between those deliverables and the EN50128 standard [RD-1].

2.1. TESTING ACTIVITIES

Generically, there are four levels/phases of testing (unit, integration, system and acceptance), each one executed during distinct phases of the project life cycle. Also, each level has assigned different roles and responsibilities for its execution according to the project. The Test Team is responsible for both SW Integration Tests (2.3) and System Tests (2.4), while the Development Team is responsible for the Unit/Module Tests (2.2).

Acceptance tests were included here for completeness.

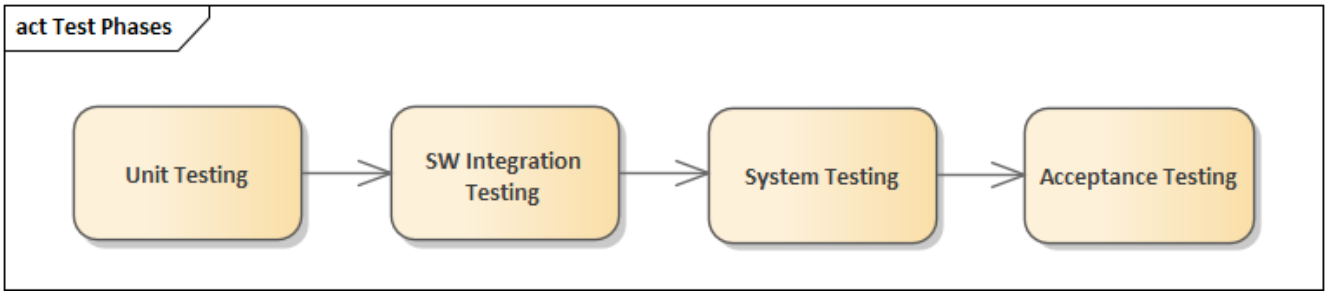

FIGURE 2: Project Test Phases

The unit/module tests searches for defects and verifies the functioning of Software bits that are separately testable.

The SW integration testing is the phase of software testing in which individual software modules are combined and tested as a group. The main testing goal is to detect problems in the module's interfaces and modules interactions. Generically, integration testing use, as inputs, modules that have been tested by unit/module testing, groups them in larger aggregates, applies tests defined in test specifications to those aggregates, and delivers as output the integrated system ready for system testing.

After the conclusion of software integration testing (2.3), the software will be tested on a complete system (2.4) to evaluate the compliance with the system requirements.

2.2. UNIT TESTING

The unit/module testing is a method of testing the correctness of a particular module of source code. The aim is to write test cases for every non-trivial function or method in the module so that each test case is independent from the others if possible. Therefore, the aim of unit testing is to verify the correct behaviour of each component of the program independently of each other. This test phase helps to eliminate uncertainty in the pieces themselves. By testing the parts of a program first and then testing the sum of its parts will make integration testing easier.

2.3. SOFTWARE INTEGRATION LEVEL TESTING

The SW integration tests validate the PROJECT_SR as a SW application. Enclosed in the PROJECT_SR, there are several SW sub-components that not only provide the main functionalities to interface with the external systems but also manage, control and maintain the PROJECT_SR component behaviour. The purpose of executing these SW tests is to validate the PROJECT_SR and all its individual sub-components as a group, and also to verify the software architecture decisions, described under SW Architecture Specification (SAS), are suitable and fulfil the user requirements (UR).

This testing phase is initiated following the unit/module tests completion and proves the SW is suitable for system testing. A comprehensive analysis needs to be performed when specifying the SW test cases to avoid overlapping with already existing unit/module test cases.

Summing up, the SW level tests will focus on proving the suitability of the SW architecture and showing that each SW component provides the specified interfaces for the other ones, by executing the components together (SW integration), while using the simulation environment (WP3.3 in FIGURE 1).

2.4. SYSTEM LEVEL TESTING

The system testing validates the PROJECT_SR as an integrated component of a whole system. The purpose of executing the system tests is to validate the complete end-to-end system, with the external interfaces. Therefore, the external interfaces of the PROJECT_SR are a big part of the scope of this level of testing as well as the target environment characteristics, such as the HW and OS dependencies. This implies the need of integration with the actual HW to fully cover the system tests. This HW/SW integration is responsible for addressing and show that the PROJECT_SR runs in a proper way on the HW, using the hardware interfaces accordingly and meeting the required timings (UR).

These system tests will target the completed PROJECT_SR SW and might be conducted, in a first stage, using the simulation-based environment. The simulators represent the external system behaviours at the interface level. Then, the HW integration is accomplished when testing the completed PROJECT_SR SW in the target platform.

Summing up, system level tests will focus on overall PROJECT_SR SW test while using both the simulation (WP3.3 in FIGURE 1) and target environments (WP3.4 in FIGURE 1).

2.5. REQUIREMENTS COVERAGE

2.5.1. Features to be tested

It is intended to cover all the system and software functionalities detailed under the System Requirements Specification (SyRS).

In other words, this means the set of tests must cover the system requirements recognised as testable. The 100% coverage of the system requirements is accomplished with a combination of tests and, if necessary, manual inspections. Each of the specified requirements is validated by, at least, one validation test case or an analysis of the developed process artefacts.

It is also intended to cover a specific set of requirements under the SwRS and decisions under the Software Architecture Specification (SAS) documents. The unit/module tests will, in the first place, cover most of those requirements. The remaining SW requirements will be covered via SW integration testing or code analysis. Therefore, the 100% coverage of the software requirements is accomplished with a combination of unit, SW integration tests and code analysis.

2.5.2. Features not to be tested

Some requirements are not fully testable within the test environment. For instance, any existing performance or non-functional requirements where response time or latency (delays) matter, cannot rely on the computer-based simulators in use. Consequently, those results do not provide official evidence of the PROJECT_SR operation in the real environment.

All the requirements that are not appropriate to be tested, either due to the reasons mentioned above or simply because are they only fit for a specific testing environment, are individually captured below with a rationale detailed for this decision (TBC).

REQUIREMENT	JUSTIFICATION
REQ_ID	The PROJECT_SR developing language can be assessed by inspection. Not testable.
REQ_ID	The storage of the CRC in the main memory shall be verified during unit/module tests. Main memory is not accessible in SW integration or System testing.
REQ_ID	Through testing we can guarantee communication between internal modules and the external environment is functional. However, it is not possible to test the internal modules are using JMS since this is a SW API.
REQ_ID	The SIL level of the PROJECT_SR is achieved by means of following a development process according to the standards and qualitatively demonstrating the achieved SIL2 THR.
REQ_ID	The safety management activities are validated through analysis and/or inspection.
REQ_ID	

TABLE 5: Requirements not covered by tests

Each one of the indicated requirements, that is not validated by any test case, shall be validated by analysis of development process artefacts. Some of these requirements can be possibly covered during unit testing phase.

Each requirement listed under TABLE 6 (TBC), is expected to have one or more test cases specified. However, due to the given justification, those test cases do not entirely cover the requirement content. This means that to obtain full coverage an additional static analysis or inspection is required.

REQUIREMENT	JUSTIFICATION
REQ_ID	This requirement foresees the need for the OS to be available. There is a risk that the application behaviour is different and that some unexpected errors occur. It should also be tested while integrating the PROJECT_SR in the target system (real environment).
REQ_ID	The negative test of this requirement, by forcing a memory corruption and thus detecting a CRC error will be covered by unit/module testing and code inspection.
REQ_ID	Communication with external system X using SSL/TLS configuration is not yet supported.
REQ_ID	The overflow situation cannot be externally tested, since the sequence number is generated randomly by the PROJECT_SR. Furthermore, the sequence number is a 32-bit number which makes it unfeasible to perform an error and trial test.
REQ_ID	

TABLE 6: Requirements partially covered by tests

2.6. APPROACH

Firstly, unit/module test cases will be specified and created based on the existing SwRS and SAS and will be placed in the first document These tests are developed using Java language and are launched using Junit framework. Additionally, in order to evaluate and provide code coverage metrics, the JaCoCo library, a Maven plugin, will be used. This is a code coverage library for Java projects that offers instructions, line and branch coverage and also generates reports. Regarding the execution approach, it is expected to automatically execute these tests with each code commit.

Subsequently, SW Integration tests will be specified and created to complement the unit/module tests and thus increase requirements coverage. These test cases will be placed in the SwRTS.

Then, system test cases will be specified and created based on the existing System level requirements. Generically, at this level both expected and unexpected behaviour will be taken into consideration when designing the test cases. Then, each test case will be executed using a simulation environment, thus adequately testing the System/SW level requirements. Before the execution taking place, the test cases will be evaluated to identify the ones that are feasible for automation, which is also dependent on the simulator's features and interfaces. So, automated test scripts to be executed on the simulation environment will also be developed prior to the test execution.

As an initial approach for the simulation environment, the testing phase is planned to be done using simulators that replicate the behaviour of the systems that interact with the PROJECT_SR component. Thus, these simulators act as interface simulators:

- Simulator 1
    - Acts as External System 1;
    - ZZZ protocol.
- Simulator 2
    - Acts as WTMK2;
    - YYY protocol.

More information about these simulators, the main features used and how they will be used for this testing phase is given under the test environment sections, 2.7.1 and 2.7.2.

The PROJECT_SR component being developed will thus interact with both mentioned simulators. Based on this, a full simulation test environment is expected to be available and properly configured in order to conduct this first

testing phase. This approach allows specifying and executing test cases in a controlled environment without the hardware-specific integration issues of a real target environment. These simulators require a high level of stability and **robustness** to guarantee efficient test executions and representative results.

The environment set up comprises an individual configuration of each simulator, launching the PROJECT_SR instance(s) with the given configuration, preparing the test scripts (when applicable), in order to have a ready test scenario.

Some requirements are not able to be fully tested within this test environment. For instance, any existing performance or non-functional requirements where response time or latency (delays) matter, cannot be relied on the computer-based simulators in use. Even if the requirements are not fully tested or the results representative for system qualification, using a simulator-based environment as a first testing phase is a very valuable strategy. Not only because it reduces complexity and time consumption (HW integration) but also because it brings a higher confidence level before loading the SW on the target system.

The simulators to be used during the first testing stage are expected to have some ability to allow a certain automation degree via scripting. This possibility will be explored and used whenever possible since it guarantees reproducibility and increases the test speed. On the Simulator 1 side, scripts can be used to create an automated test suite. For the Simulator 2, it is expected to have more manual tests (TBD). However, the decision depends on the test case and on the existing simulation environment limitations, that need to be evaluated case by case.

The PROJECT_SR component, which is the element under test, needs a set of configuration data to be provided. To conduct all test cases, multiple configuration sets are going to be used. This is required to support specific test scenarios. However, for most of the functional tests, a standard configuration set will be enough.

Several test techniques and methods are planned to be used, ensuring the features of the system are adequately tested. These techniques are outlined in the QAP following the guidelines of the EN50128 standard [RD-1]. However, for completeness and for a better understanding those are further explored here. The PROJECT_SR set of features will be tested mainly with Functional/Black-box and Dynamic SW Analysis testing techniques. Given the nature of the PROJECT_SR system and its requirements, test case execution from boundary value analysis as well as equivalence classes and input partitioning represent the core of the testing methods.

TECHNIQUE	NOTES
Test Case Execution from Boundary Value Analysis	Test Cases covering the limits and boundaries of input range
Equivalence Classes and Input Partition Testing	Test Cases covering each partition of the input domain
Performance Testing	Stress and memory constraints testing

TABLE 7: Summary of testing techniques

Specific situations where System level or SW Integration test cases are not capable of externally driving the SW to trigger and detect error situations, can be overcome by the use of JDB to change the program behaviour, namely insert errors. This is particularly useful since it does not require an unofficial build of the PROJECT_SR with the required error types injected to be compiled.

Specific performance tests should also be considered and executed while in the simulation-based environment. These include load and stress tests as well as memory constraints tests. However, timing response related tests are not considered here mainly because simulators are not accurate enough to allow obtaining meaningful results. Those should be addressed while integrating the PROJECT_SR in the target HW and with the external systems.

2.7. TEST ENVIRONMENT

The overall diagram of the expected complete and full test environment is given below. It describes the behaviour and interactions between the tester and the systems involved in the test environment.

Test Environment Figure

FIGURE 3: Test Environment

2.7.1. Simulator 1

TBC.

2.7.2. Simulator 2

TBC.

2.7.3. Network Links

TBC.

2.7.4. Environment needs

Regarding the simulated environment, both Simulator 1 and Simulator 2 will be installed and run in a VM that fulfils the minimum system requirements stated by both simulator manuals. On the other hand, each PROJECT_SR instance will run on individual VMs. These PROJECT_SR VMs shall replicate, as close as possible, the characteristics of the PROJECT_SR HW platform, namely main memory, storage and processing power.

It is expected that each VM have network connectivity capabilities and are configured in order to be reachable via network from the other VMs.

ETC.

2.8. TEST ARTEFACTS

A Test Plan should be available and issued (this document section), prior to conducting the testing activities.

The System and SW Test Case Specifications produced with enough detail are mandatory. The requirements specifications are the baseline for test cases definition and execution. A high-level definition of the test objective and functionalities covered is expected to be provided. Also, a list of test cases is expected to be defined here. Each test case partially or completely covers a requirement or a set of requirements. The test cases can be elaborated in MS Word or Excel formats, and shall have the following elements/attributes included in TABLE 8.

ATTRIBUTE	DESCRIPTION
Identifier	Unique Test Case Identifier: SYS_TCS_MMM_NNNN
Test Case Execution Responsible	Entity responsible for executing the test case.
Purpose	High-level description of the test case, containing what is intended to be tested and achieved.
Author(s)	Team responsible for elaborating the test case.
Precondition(s)	List/Description of conditions that should be met prior to performing the test (e.g. initial state of system or particular test cases that shall be passed first).
Input Specifications	List of test steps.
Output Specification	List of expected results for each test step, when applicable.
Notes	Other information related to the test case, such as further details, clarifications or relationships with other test cases.

TABLE 8: Elements of the Test Cases

The Purpose element shall provide a brief description of the test case and its objectives.

The Preconditions shall describe the state the system should be in before the test is executed. The description should fully define the desired start state. Certain conditions and needs that should be specifically met for the simulation environment shall be considered here as well.

The Input Specifications shall describe the inputs to be provided to the test, as stimulus or actions towards the PROJECT_SR component. This can be organized as test steps to be sequentially executed or input to the system under test. The inputs shall clearly define the value or range of values admissible when applicable.

Finally, the Output Specifications shall identify the overall expected test result as well as the expected output for each test step individually, when appropriate. This also comprises detailing how these outputs are verified, such as examining log files or particular files or screens.

The test cases shall be organized in a table-like format, such as the one presented below:

TEST CASE IDENTIFIER:	<>
Test Case Execution Responsible:	<>
Purpose:	<>
Author(s):	<>
Precondition(s)	<>
Input Specifications	<>
Output Specifications	<>
Notes	<>

TABLE 9: Test case template

Whenever possible, the test cases will be written as generic as possible, in a way that the content applies for both simulation and target environment tests. Any particular details concerning the environment setup should be clearly identified on the test case itself (precondition or the particular test step), so the tester or integrator knows when some step applies only to simulation or real environment only.

A traceability matrix that maps each requirement to the corresponding test case(s) covering it shall be created. Tests are derived from the specifications, either requirements and architecture, and must be traced back to them. The objective of this traceability is to ensure that all requirements can be shown to have been properly covered and that no untraceable material exists. In the end, each requirement should be met by at least one test case.

The process of designing the test cases shall be complemented with the creation of Test Scripts. Each test case shall be evaluated, and whenever feasible, automated scripts should be applied. These automated test scripts are primarily intended to be run on the simulation environment. Automated test scripts should include two major sections. One that designs and/or verifies the test case precondition, and another one that clearly separates each test step and within each test step each action and expected outcome. On the other hand, manual test scripts need to precisely define every operator action and expected outcome, following the same structure and order of the test case.

System/SW and Module test reports are the documents that describe the results of the tests carried out. Therefore, they should include a test log for recording the test results and the faults identified during the test runs. Moreover, the identity and configuration of all items involved (hardware, software or simulators used, test scripts as well as version information of the test specification) shall be documented. In the end, an evaluation of the test coverage and test completion shall be provided, and any deviations noted. Ideally, every test report should have some automation to be easily filled and gather the results. Further manual processing may be needed, such as

individual failure analysis. Also, the test report should be simple and easy to fill in. As a guideline, the report contains the following main sections:

- Environment Definition;
- Test Execution;
- Test Log;
- Faults Register;
- Summary Report.

2.9. TEST DELIVERABLES

A list of the client deliverables for the project, regarding every testing phase, is presented in the table below (TABLE 10). These deliverables comprise the Test Specifications and Test Reports items that are detailed in the above section (2.6).

REF.	DESCRIPTION	DELIVERABLE ID	MILESTONE
	Module Test Report		
	System Requirements Test Specifications		
	SW Requirements Test Specification		
	System Requirements Test Report		
	Software Requirements Test Report		
	Test Scripts		
	Test Logs		

TABLE 10: List of testing deliverables

According to the EN50128, the Overall SW Test Specification shall contain a description of the tests to be performed on the completed SW. The techniques and measures to be used are detailed under the Approach section (2.6). The SW Integration Test Specification shall address the behaviour of the SW components when executed together. The understanding of System and SW level testing (2.2 and 2.4) is similar and matches to what is expected to be covered in each of these specifications.

2.10. TEST CRITERIA

The following table (TABLE 11) outlines start, end and exit key criteria of unit/module test phases.

CRITERIA	DESCRIPTION
Test Start	Test plan, SW Requirement Test Spec and design documents are released; PROJECT_SR build available; Test cases and test scripts developed and under configuration control.
Test End	100% test execution is achieved.
Test Exit	Test records issued and failures documented; All test outputs and deliverables issued (2.9 Test deliverables).

TABLE 11: Test Criteria for Module Tests

The following table (TABLE 12) outlines start, end and exit key criteria of SW Integration test phases.

CRITERIA	DESCRIPTION
Test Start	Test plan, SW Requirement Test Spec and design documents are released; Testing environment setup ready; PROJECT_SR build available; Test cases and test scripts developed and under configuration control. Previous test phase successfully completed (Module Test Report available and approved).
Test End	100% test execution is achieved.
Test Exit	Test records issued and failures documented; All test outputs and deliverables issued (2.9 Test deliverables).

TABLE 12: Test Criteria for SW Integration Tests

The following table (TABLE 13) outlines start, end and exit key criteria of System test phase.

CRITERIA	DESCRIPTION
Test Start	Test plan, System Requirement Test Spec and design documents are released; Testing environment setup ready; PROJECT_SR build available; Test cases and test scripts developed and under configuration control. Previous test phase successfully completed (SW Integration Test Report available and approved).
Test End	100% test execution is achieved.
Test Exit	Test records issued and failures documented; All test outputs and deliverables issued (2.9 Test deliverables).

TABLE 13: Test Criteria for System Tests

As introduced earlier, the testing activities heavily rely on a simulation-based environment. Hence, before starting the test execution, it is expected an existing stable environment and properly configured simulators. Any major instability or erroneous behaviour on the simulators side may compromise the validity of the test results, and so the testing activities shall be suspended in that case. Testing activities can be resumed as soon as the test environment is properly set up again and the required adjustments have been made on the simulators, such as the deployment of a new version or providing a different configuration.

It is also important to mention that, apart from the overall maturity level, the simulators may lack some functionalities that prevent some test cases from being executed. Shall this happen, the possibility of covering those test cases with inspections or through SW Integration tests needs to be evaluated. The testing activity may continue as long as the affected and dependent test cases are skipped.

When a testing activity starts, for instance, system testing or software testing, a full test execution should be made while maintaining the same simulators version. If a new version or configuration of any simulator is provided, the tests shall be re-executed to assure consistent test results.

2.11. ITEM SUCCESS/FAILURE CRITERIA

Each created test case should identify the criteria to be used for evaluating and determining the success or failure of that test case. More specifically, each test case should detail the expected outcomes and results like explained and described in the test case skeleton (TABLE 8). Note that, if the test case has multiple test steps, intermediate expected results should also be presented.

Moreover, as previously stated, the success criteria of the phases of System and SW testing is established as every test case being executed with success (100% test execution) without any problems/issues detected. This means that every test case should be executed and successfully passed.

2.11.1. Problem classification

The test team members, with testing responsibilities, should be aware of the existence of different problems that have different levels of impact on the system. There are faults that compromise the entire system and are therefore not acceptable; there are other faults of less importance that do not invalidate the functionality being tested.

The project team, when testing, should classify identified problems based on the following scale, shown in TABLE 14.

CLASSIFICATION	DESCRIPTION
Severe	Failure in parts of the SW component
Medium	Minor impact or interference to the SW component
Minor	No function impact on the SW component

TABLE 14: Faults level classification

These criteria are subjective, therefore when in doubt the responsible for the execution of the test cases, SPAE and/or PM should decide which level is the most appropriate. Additionally, the agreed classification level further needs to be accepted by the client.

2.11.2. Problem reporting

During the test activities, namely the execution of the test cases, the different issues (problems) identified should be registered, documented and maintained. Jira, an issue tracking system, will be used to keep track of those faults. The reporting procedure should include the problem description, the classification based on the impact it poses to the system, as described above in 2.11.1, and any mitigation actions identified. This procedure should be suitably documented in the System/SW test log, which is a part of the Test Report, and provided to the client as an “PROJECT_SR Issue Tracker” and delivered following the defined process.

2.12. RESPONSIBILITIES

The project organization is compliant with the level of independence to meet the SIL2 requirements of the standards. The roles and persons assigned to each role associated with the testing activities can also be found in TBC.

2.13. STAFFING AND TRAINING NEEDS

Training needs are identified in TBC.

2.14. SCHEDULE

The testing activities will be under the control and monitoring of the master schedule. The complete schedule is provided in TBC.

2.15. RISKS AND CONTINGENCIES

The risks and corresponding mitigations were identified in an early stage of the project and are placed down in TBC. Currently, the ones that apply to the test plan are shown in the table below. The risk identification process should be continuously performed, for instance, during project meetings or as the testing activities start and progress. A certain degree of uncertainty exists, based on current knowledge of the simulators and the overall test environment. Thus, new risks may arise and effective actions to overcome them need to be placed.

RSK__NBR	
RISK NAME	**Bla Bla Bla**
RISK DESCRIPTION	Risk Description
RISK MITIGATION	Mitigation strategies and actions.

TABLE 15: Testing validation risks

2.16. TEST CASE EXAMPLE

TEST CASE IDENTIFIER:	TST_ID_NBR
Responsibility	Tester Name
Purpose	This test shall verify that the PROJECT_SR logs an error and terminates the instance, when there is a mismatch between the CRC of the configuration structure in memory and the one calculated at start-up
Author(s)	TC author
Precondition(s)	a. No log files exist. b. The PROJECT_SR is running with its default configuration. c. The External Systems instances are running with their default configurations.
Input Specifications	a. Using jdb, change the configuration CRC value initially stored. b. Run the PROJECT_SR for 60 seconds.
Output Specifications	a. Check that the PROJECT_SR logs an error indicating a mismatch between the initially stored CRC value and the one periodically calculated. b. Verify that the PROJECT_SR instance was terminated in the host, by checking that no PID exists for the PROJECT_SR process.
Note	NA