Test Plan

Version No. :

Date:

Project Name:

Project Code:

Guidance to Users of this template

1. *Text in italics within angled braces* “< >“in this document are only instructions. Delete the text in italics with the braces after completing the section /document
2. The Text in normal font is the recommended text, readily provided. User may modify the content provided it does not violate the IEC61508 specifications or CMMI guidelines.
3. Delete Subsections if not applicable for project.
4. Delete this guidance note, once project specific FSMP is created.

Copyright Notice

This document contains proprietary information of HCL Technologies Ltd. No part of this document may be reproduced, stored, copied, or transmitted in any form or by means of electronic, mechanical, photocopying or otherwise, without the express consent of HCL Technologies. This document is intended for internal circulation only and not meant for external distribution\*.

*<\* - Remove the last sentence. That is intended for template*>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version No | Date | Prepared by / Modified by | Significant Changes |
|  |  |  |  |
|  |  |  |  |

Reference Documents

< List all supporting documents that are referenced in this document >

| Document Name | Version /Date | Location |
| --- | --- | --- |
| IEC61508-3 | Ed2, 2010 |  |
| IEC61508-7 | Ed2, 2010 |  |
| <Functional Safety Management Plan > | <latest version/date> |  |
| <Safety Requirement Specification > | <latest version/date> |  |
| <Build Plan > | <latest version/date> |  |
| <Tools Qualification> | <latest version/date> |  |

Table of Contents

[1 Definitions, Acronyms and Abbreviations 5](#_Toc312080290)

[2 Introduction 6](#_Toc312080291)

[2.1 Objective of this document 6](#_Toc312080292)

[2.2 Project Summary 6](#_Toc312080293)

[2.3 Test Scope 6](#_Toc312080294)

[2.3.1 Inclusions 6](#_Toc312080295)

[2.3.2 Exclusions 6](#_Toc312080296)

[2.3.3 Test Assumptions 7](#_Toc312080297)

[2.3.4 User Acceptance Criteria 7](#_Toc312080298)

[3 Test Team 7](#_Toc312080299)

[4 Testing Methods 9](#_Toc312080300)

[4.1 Test Planning 10](#_Toc312080301)

[4.2 Test case / Test Script Development 11](#_Toc312080302)

[4.2.1 Test Script 13](#_Toc312080303)

[4.3 Test Environment 13](#_Toc312080304)

[4.4 Test Execution 13](#_Toc312080305)

[4.4.1 Impact Analysis 15](#_Toc312080306)

[4.4.2 Regression Testing or Re Verification 15](#_Toc312080307)

[4.4.3 Defect Severity 15](#_Toc312080308)

[4.4.4 Assessor’s Role during Test Execution 16](#_Toc312080309)

[4.4.5 Automation Test Case Execution 16](#_Toc312080310)

[4.5 Test Results 16](#_Toc312080311)

[5 Test Execution Strategy 17](#_Toc312080312)

[5.1 Test Coverage - SIL Level Wise 17](#_Toc312080313)

[5.1.1 Module Testing 18](#_Toc312080314)

[5.1.2 S/w Integration testing 18](#_Toc312080315)

[5.1.3 H/w – S/w Integration testing 19](#_Toc312080316)

[5.2 Additional Test Techniques 19](#_Toc312080317)

[5.3 Testing Techniques for Safety Requirements 20](#_Toc312080318)

[6 Code Coverage 21](#_Toc312080319)

[6.1 Coverage Criteria 21](#_Toc312080320)

[7 Test Automation 22](#_Toc312080321)

[7.1 Summary 22](#_Toc312080322)

[7.2 Modules to be Test Automated 22](#_Toc312080323)

[8 Verification & Validation Tools 22](#_Toc312080324)

[9 Test Environment Setup 23](#_Toc312080325)

[9.1 Test Equipments 24](#_Toc312080326)

[10 Defect Management 25](#_Toc312080327)

[10.1 Responsibility 25](#_Toc312080328)

[10.2 Defect Logging 25](#_Toc312080329)

[10.3 Defect Analysis 25](#_Toc312080330)

[10.4 Defect Status Transition 25](#_Toc312080331)

[10.5 Defect Meetings 27](#_Toc312080332)

[10.6 Defect Prevention Plan 27](#_Toc312080333)

[11 Suspension and Resumption Requirements 27](#_Toc312080334)

[11.1 Suspension 27](#_Toc312080335)

[11.2 Resumption 27](#_Toc312080336)

[12 Software Metrics 27](#_Toc312080337)

[13 Traceability to IEC61508 28](#_Toc312080338)

[13.1 Mapping to Specifications 28](#_Toc312080339)

[13.2 Mapping to Testing Techniques in IEC61508-3 Annexure A & B 33](#_Toc312080340)

# 

# Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Description |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

## Objective of this document

*<This document describes the test plan for Unit Testing, Integration and System testing. It includes*

* *The team composition*
* *Test Strategy*
* *Test Environment*
* *Test tools*
* *Test Pass /Fail Criteria*
* *Defect Management>*

## Project Summary

*<Give a brief overview of the Project, with reference to domain, technology, problem statement and type of the project>*

*<Also give a brief summary of the product / application to be tested. The high level functionality of the application should be given.>*

## Test Scope

*<General application scope should be provided in this section. >*

### Inclusions

*<Module wise detailed inclusions should be tabled here. >*

### Exclusions

*<Exclusions refer to modules that are part of overall system but not tested as per this plan. It could include*

* *Modules that could be tested only at End User environment. In such cases, the following table should be filled*

| Modules that cannot be tested within test Environment | Environment Required to test Module | When /where will Module be tested |
| --- | --- | --- |
|  | For example the test case could require manipulation of keys in hot/cold chamber  Test could require Peer to Peer communications with another controller not made available during testing |  |
|  |  |  |
|  |  |  |

* *System validation testing is not part of Project scope, then it should be mentioned here>*

### Test Assumptions

*<Any assumptions related to testing should be mentioned here. For Example, if Customer is responsible for delivering a software component, then the Assumption is that component will be fully tested by the customer before used for Integration testing. >*

### User Acceptance Criteria

*<Indentify the key performance parameters that necessarily need to pass. This will be objective or goal of entre testing exercise. The Key performance parameters would include functional and non functional requirements.*

*For example, the testing is considered to be successful provided*

* + *All indentified Major Defect are resolved*
  + *The Performance criteria of loop was within the specified limits ( < xxx seconds)*
  + *The software functioned without any error or warning when operated continuously for 72 hours*

*Reference should exist in SRS on User acceptance criteria.>*

# Test Team

| # | Name | Project Role | Responsibilities | Contact Details |
| --- | --- | --- | --- | --- |
|  |  | Project Manager | Review test report , Analyze defects and classify them as Invalid, Valid or specification issue |  |
|  |  | Test Manager\* | Prepare test Plan, Review Test cases, Configure Defect Tracking Tool for Project, Analyze defects, Resource planning, tracking testing progress. |  |
|  |  | Test Lead | Prepare Test cases , Set up Test bed, , Prepare Test Summary Report |  |
|  |  | Test Engineer | Execute Test cases, Log Defects, Perform regression testing |  |
|  |  | Developer | Module (or Unit) test case generation and Execution, Prepare test report |  |

*Note- It is normal for the project manager to play the role of test manager. However for large projects (for example test team > 10 resources), there shall be dedicated Test manager. The Decision to have dedicated Test Manager will be governed by complexity and size of test team and commercial implications*

*Integration Testing will be done by a separate independent team. Test Case execution will be done by Test engineers and Test Case generation by Test lead.>*

# Testing Strategy

*<This section briefs the software testing methods used to verify the software.*

*Testing provides a method of verification (in the form of test cases) to ensure compliance with a software requirement by a thorough exercising of the unit under appropriate conditions in accordance with documented procedures.*

*Three types of testing and its objectives are listed below.*

* *Hardware/software integration testing: To verify* 
  + *Correct operation of the software in the target computer environment.*
  + *All requirements , including safety requirements , are satisfied*
* *Software integration testing: To verify* 
  + *the interrelationships between software requirements and components and*
  + *The implementation of the software requirements and software components within the software architecture.*
* *Unit or Module level testing: To verify*
  + *The implementation of software low-level design.*

*Testing Phase consists of following Major activities:*

1. *Test Plan Creation*
2. *Test Case and Procedure Development*
3. *Test Environment Set-up*
4. *Test Execution*
5. *Test Results*

*Each phase is explained below >*



## 

## Test Planning

| Entry Criteria | Tasks | Verification | Exit Criteria | Deliverables | Document Owner ,Author |
| --- | --- | --- | --- | --- | --- |
| Approved FSM plan  Approved SRS | Test Plan Preparation following the template | Test Plan checklist | Internal Approved Test Plan | Test Plan | PM, PM[[1]](#footnote-1) |

*<Test Plan (this document) shall be Prepared after the Approval of SRS. Test Plan shall need to be approved by assessor. The Assessor is defined in Functional safety management Plan>*

## Test case / Test Script Development

| Entry Criteria | Tasks | Verification | Exit Criteria | Deliverables | Document Owner ,Author |
| --- | --- | --- | --- | --- | --- |
| S/w Integration test Case Development | | | | | |
| Test Plan  Approved SRS  Approved HLD  Build Plan | S/w Integration Test Case creation | Internal reviews,  Test execution Checklist | Review and Approval of Test cases | Approved S/w Integration Test Cases  Complete Relevant columns of traceability Matrix | PM, Test Lead |
| S/w –H/w Integration test Case Development | | | | | |
| Approved SRS  Approved HLD  S/w Intg Test Cases  S/w-H/w Build Plan | S/w –H/w Integration Test Case creation | Internal reviews, checklist  Test execution Checklist | Review and Approval of Customer | Approved S/w-H/w Integration Test cases  Complete Relevant columns of traceability Matrix | PM, Test Lead |
| Module Test Case Development | | | | | |
| Approved LLD | Module /Unit Level Test cases creation | Internal reviews, checklist  Test execution Checklist | Review and Approval of unit test cases | Approved Unit Test cases | Tech Lead, Developer |

*<A test case is a single set of inputs, initial conditions and expected results to verify fully or some aspects(s) of a high level requirements or software design. Any specific pass/fail criteria are explained in the Test Case as Expected Results. In most instances, multiple test cases will be required to provide full test coverage of a particular requirement. Templates required for generating test cases are provided in section below. Ensure sufficient Test cases are generated using the following techniques*

* *Boundary Value analysis*
* *Equivalence partitioning*
* *Performance testing*
* *Stress testing*
* *Security Testing*
* *Error Guessing (Negative testing)*

*All test cases shall be safety classified, as done for requirements into*

* *Safety Function – Functions that directly involved in safety of EUC. Failure of function on PE system will have an adverse impact on EUC*
* *Safety Support Function – Functions that do not directly impact the safety of EUC but could eventually lead to failure of safety function*
* *Safety Integrity Function – Functions that assist in determining the safety integrity of function*
* *Non Safety Function – They do not impact the safety of EUC*

*For S/w integration testing and S/w-H/w integration testing, the Build Plan document shall be one of input documents to prepare test cases. Build Plan shall describe*

* *Build responsibilities*
* *Configuration Considerations - make file, linker , compiler options*
* *Modules to be integrated*
* *Integration Sequence and Build instructions*

*The H/w-S/w integration testing shall have adequate test cases for:*

* *Each Requirement as specified in SRS*
* *Integration with all types of input sensors*
* *Performance Testing , Stress testing or Load Testing*
* *Negative Testing – Operating the equipment in unconventional way (similar to Error guessing techniques)*
* *For all modes of Equipment Under Control (EUC) and for all states of PE (Programmable Electronics) system>*

### Test Script

*<For projects where Test Automation is to be implemented, Test script generation shall also be done in parallel to test case generation. Test cases for automation shall be indentified from test case document. >*

| Entry Criteria | Tasks | Verification | Exit Criteria | Deliverables | Responsible Person |
| --- | --- | --- | --- | --- | --- |
| Test Case Document | Creating Test scripts using automation tool | Test script checklist | Completion of script for all indentified test cases | Approved Test scripts | Test Lead |

*<Note test script modification will be on going activity till final release as scripts are most likely to throw errors due to modifications to application code>*

## Test Environment

| Entry Criteria | Tasks | Verification | Exit Criteria | Deliverables | Responsible Person |
| --- | --- | --- | --- | --- | --- |
| Test Plan  Availability of Required Hardware & Software and its Installation manuals | Setup the required test bed | Verification of test environment using Zero day checklist | Implementation of all checkpoints in Zero day checklist | Test Environment Setup Document / Manual | Test Lead |

*<The Above table is generic and could be modified for Unit and Integration testing. Also refer to Guidelines on more information on setting up of test bed>*

## Test Execution

| Entry Criteria | Tasks | Verification | Release /Exit Criteria | Deliverables | Responsible Person |
| --- | --- | --- | --- | --- | --- |
| Module Testing | | | | | |
| Code has been reviewed successfully  Approved Module Test Case document  Testing Tools Calibration records | Execute Test cases | Manual Verification of filled in Test Case document | If All Unit test cases for the program are executed once and indentified defects fixed | Test Report | Test Engineers |
| S/w Integration test | | | | | |
| All the units, which share the interface, have been unit tested.  Unit test plan exit criteria met | Execute Test cases | Manual Verification of filled in Test Case document | All Integration test cases executed and No Major defect are pending | S/w Integration Test report |  |
| S/w –H/w Integration test | | | | | |
| All the interfaces between the units have been successfully tested.  Approved S/w –H/w Intg. Test case document | Execute test cases | Manual Verification of filled in Test Case document | All Test cases executed at least once and no major defect found.  All non-testable scenarios and requirements are communicated to the customer | S/w Integration Test report | PM, Test lead |

*<Test Execution will be done by test engineers and shall log the defects /defects in PMSmart (Defect management Tool). PM will analyze & classify the logged defect as Valid defect /Invalid defect (i.e.) Reject / or a specification issue. PM shall also decide on severity of defect. All safety related defects shall necessarily be classified as Major*

*For those test cases that were not executed, Test engineer shall provide justification for not executing the test case in Test case and Test Report document. The Author of test case document (Test lead / Test manager) shall provide replacement test cases and update the test case document.>*

### Impact Analysis

*<For all valid defects, an impact analysis will be done before modifying the code. The impact analysis is done by filling the Test Incident Report. In reality, all the columns of Test Incident Report will be incorporated in PMSmart (HCL defect management Tool). Impact Analysis will be done by Tech lead in association with developers.*

*For all valid defects, the assigned developer shall fix the defect and release a new executable for testing. The test cycle will be repeated till the test cases are executed successfully. For all unresolved defects, developers to state justification to not resolve them.>*

### Regression Testing or Re Verification

*<Regression testing will be carried at all levels of testing to ensure that changes (introduced as a result of resolving defects or due to requirement changes) do not introduce unintended behavior or additional errors.*

*For every valid defect, Test Incident Report indentifies Test Cases to be executed to check the resolution. During regression testing, those test cases shall be executed.*

*Any new requirements that gets added ( either due to late additions by customer or due to defect analysis ) during the testing phase are also tested as part of regression testing.>*

### Defect Severity

*<The classification of Major /Minor/Trivial defect is as follows*

| Severity | Impact |
| --- | --- |
| Major | System inoperable, An implementation that does not meet the requirements (or any other input document), Defects that may cause the system to hang, crash; produce incorrect/ unexpected results or behavior, or corrupt user data with no known work around.  All Safety Function , Safety Support Function and Safety integrity Function test defects shall have to classified as Major |
| Minor | Defects that cause incorrect results or behavior with known work around. Large and/or critical portion of the system is unaffected and would not cause operational failure. System can continue to be completely functional with that defect.  This classification is applicable only Non safety functions |
| Trivial | Defects that affect limited areas of functionality that either can be work around or ignored.  This Classification is applicable only for Non Safety function |

*During the test case execution, any modifications to System requirements will be dealt by change control mechanisms as described in Functional Safety Management Plan.>*

### Assessor’s Role during Test Execution

*<The Assessor (as defined in Functional Safety management plan) may or may not be present during the actual test execution. This will be based on commercial terms of project. Assessor’s Role, if required during Test execution shall be defined here >*

### Automation Test Case Execution

*<Automation test case scripts shall be executed for created test scripts. Test Automation is generally executed for each build created>*

| Entry Criteria | Tasks | Verification | Exit Criteria | Deliverables | Responsible Person |
| --- | --- | --- | --- | --- | --- |
| Verified Test Scripts | Execution of Test scripts | Analysis of Test report generated by Automation Tool | Error Free Execution of test scripts\* | Automation Test report | Test Engineer |

*\* - Error Free execution of test scripts is not be mistaken for Test Pass /Fail*

## Test Results

*<Test Results are published after the end of test case execution. This would include the Manual, Automation test results and also the Code Coverage Analysis. Test Results shall be reviewed by Project Manager and approved before the release of Product. User Acceptance criteria defined in requirement section shall be checked to verify it has passed the test cases.>*

| Entry Criteria | Tasks | Verification | Exit Criteria | Deliverables | Document Owner ,Author |
| --- | --- | --- | --- | --- | --- |
| Completion of Test case execution ( Module , S/w Intg test, S/w-H/w Intg Testing)  Calibration records of test tools | Prepare Test Summary report | Test Result Review checklist | Completion of document | Test Report (Module, S/w Intg test and S/w –H/w Intg. Test report ) | PM, Test Lead |
| Completion of Code Coverage testing | Prepare Test Summary report | Test Result Review checklist | Completion of document | Code Coverage Analysis (included in Test Report) | PM, Tech Lead |

# Test Execution Strategy

*< Broadly classify the type of independent testing to be done . Example:*

* *Manual Black box testing – Functional and Non Functional testing. Functionality of product and Non Functional testing include Performance, Stress , Installation Testing*
* *Automated Testing*
* *White Box testing – Mainly Code Coverage Testing >*

## Test Coverage - SIL Level Wise

This section describes, for each SIL level , the minimum testing techniques and a brief test coverage that need to be executed as part of Module testing, S/w integration testing and S/w-H/w integration testing.

*<The Description Column briefly describes the major test that will be covered using the technique. The given information under the description column is only ane xanple and user is required to define project specific information based on example provided.*

*‘HR’ – Highly Recommended-almost equivalent to mandatory and ‘R’ – Recommended(optional) and “—“ does not matter .*

*In the table below, The user should delete the SIL level columns not applicable for the project > ‘*

### Module Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Generic Method | Specific Technique | SIL Level | | | Description |
| 1 | 2 | 3 |
| Functional and black box testing: | Equivalent class testing for external and Internal interfaces | HR | HR | HR | *"Frequent build and smoke test" process are implemented.*  *Entry & Exit point Testing.*  *Expected 100% coverage for each module* |
| State Machine testing | R | HR | HR | *Test coverage of different states of a state machine.* |
| Boundary Value Analysis | R | HR | HR | *Test for inputs at the low and high range limits, possible range limits (for example for Analog input signal in range of 4..20mA , send 0mA,4mA, 20mA and 25mA to check software functionality)* |
| Dynamic Analysis and testing | Code Coverage testing | -- | -- | HR | Refer to Section 6 : Code Coverage |
| Code Review on Code not reached by Code coverage testing | -- | -- | HR |  |
|  |  |  |  |  |  |

### S/w Integration testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Generic Method | Specific Technique | SIL Level | | | Description |
| 1 | 2 | 3 |
| Functional and black box testing | Test using the equivalence classes for the external Interfaces | HR | HR | HR | *Test the performance consistency of the system with different interfacing modules* |
| State Machine testing | R | HR | HR | Covering Individual functionality of every state machine used in software modules and associated with respective hardware. Coverage of normal and critical test failure scenarios |
| Model Based testing | R | R | HR | Usage of Finite State machines / Decision tables /State Charts /Markov Chain Models to generate test cases |
| Dynamic Analysis and testing | error guessing | -- | HR | HR | Test coverage of all positive and negative test cases by fault attack techniques  Use SCA/Hazop analysis for generating test cases |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

### H/w – S/w Integration testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Generic Method | Specific Technique | SIL Level | | | Description |
| 1 | 2 | 3 |
| Functional and black box testing | Test using the equivalence classes for the external Interfaces | HR | HR | HR | Entire Product functionality testing |
| Boundary Value Analysis | HR | HR | HR |  |
| Model Based testing | R | R | HR | *Usage of Finite State machines / Decision tables /State Charts /Markov Chain Models to generate test cases* |
| Performance testing | Performance and Response time testing | HR | HR | HR |  |
| Stress testing | -- | -- | HR |  |
|  |  |  |  |  |  |

## Additional Test Techniques

*<To increase the test coverage, listed below are other techniques. Refer to Guidelines for more information on each of sections. In each of cell indentify if the techniques is applicable and will be tested in that phase of testing. >*

| Test Technique | Unit Testing | S/w integration testing | S/w –Hw Integration testing |
| --- | --- | --- | --- |
| User Interface test | *<Applicable / Not Applicable>* |  |  |
| Transactions Test |  |  |  |
| Capacity and Volume Test |  |  |  |
| Recovery Test |  |  |  |
| Security Test |  |  |  |
| Repetition Test |  |  |  |
| Race Condition Test |  |  |  |
| Configuration Testing |  |  |  |
| Compatibility Testing |  |  |  |
| Installation /Uninstallation testing |  |  |  |

## Testing Techniques for Safety Requirements

<Indentify testing strategy and techniques to be used to test the Safety Requirements. A Justification for choosing the techniques should also be provided. The Justification should include why other techniques are not suitable for testing the safety requirement. Note Safety requirements testing will be done as part of H/w-S/w integration testing. >

| Safety Req. ID ( from SRS ) | Testing Strategy /Techniques to be used | Tool to be used | Justification |
| --- | --- | --- | --- |
| xx-yyy-01 | Boundary value Analysis | - |  |
| xx-yyy-02 | Equivalence partitioning test |  |  |
| xx-yyy-03 | Stress testing |  |  |
|  |  |  |  |

# Code Coverage

<Code Coverage or Structured test coverage is done for all SIL levels. However the coverage required varies as per the SIL level. As per the Table B-2 IEC61508-3 specification, the following table need to satisfied for different SIL levels >

| Description | SIL1 | SIL2 | SIL3 |
| --- | --- | --- | --- |
| Entry Points -100% Coverage (also called Function Coverage) | HR | HR | HR |
| Statements – 100% Coverage | R | HR | HR |
| Branches – 100% Coverage (also called Decision Coverage) | R | R | HR |
| MC/DC – 100% Coverage | R | R | R |

## Coverage Criteria

*<Identify the Coverage Criteria required. Refer to Test Guidelines for more information on each of metric. The % Coverage required as per SIL level, as stated in IEC 61508 specifications, is mentioned in table above. In the %Coverage required column, need to retain or remove the %coverage required as per the SIL level and also include the intended coverage required for other metrics>*

| # | Coverage Metric | % Coverage Required | Coverage tracking possible with Tool?  (Yes/No) |
| --- | --- | --- | --- |
|  | Statement Coverage | 100%\* |  |
|  | Decision Coverage | 100%\* |  |
|  | Condition Coverage |  |  |
|  | Multiple Condition Coverage |  |  |
|  | Function Coverage | 100%\* |  |
|  | Modified Condition /Decision Coverage | 100%\* |  |
|  | Call Coverage |  |  |
|  | Data Flow Coverage |  |  |
|  | Loop Coverage |  |  |

\* These are requirements as per IEC61508-3 specification

The Code Coverage results shall carry justification when coverage is < 100% and also suggest techniques to test uncovered code. Also refer to guidelines on techniques to increase code coverage.

# Test Automation

## Summary

*<Briefly describe*

* *Reasons for doing Test Automation – like Customer mandated or many periodic releases envisaged or in general to improve internal resource productivity >*
* *Describe the Tool and salient features >*
* *Justification for selecting the Tool vis-a-vis other Tools – consider parameters like Compatibility the application under test, test harness ,Total Cost of the tool, Productivity, Ease of use, Learning curve, Stability of the Software tool vendor, System requirements, Supported environments and Installation*
* *Frequency of Execution - Should automated test scripts be executed for each build or every major release.*

## Modules to be Test Automated

*<Identify the Modules that are to be Test Automated>*

# Verification & Validation Tools

Testing tools are described in this section. This includes Verification and Validation Tools. The H/w and S/w required to set up the Test bed will be covered in next section on Test Environment.

Tools used in conducting Tests should be reviewed at defined intervals to determine if there is a need to improve and/or upgrade them .The effect of using such tools on the quality of the software product should be carefully considered.

*<To ensure only qualified tools (with T2 and T3 criticality) are used , reference should be provded to Tools Qualification document. If Tools Qualification document is not prepared at time of writing this document, reference should be added later >*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classification | Tool and Version | Purpose | Criticality ( see note below) | Traceability to Tools Qualification Document |
| Testing Tools ( Code Coverage) | Mccabe |  | T2 |  |
| Testing Tools (black Box testing) |  |  | T2 |  |
| Performance /Stress testing tools | NI’s LabView |  | T2 |  |
| Defect tracking | PM SMART |  | T3 |  |
| Static Analysis Tool | QA-C | Code Analysis | T2 |  |
| Test Automation | Win Runner |  | T2 |  |

Criticality:

Software off-line support tool a software tool that supports a phase of the software development lifecycle and that cannot directly influence the SRS during its run time. Software off-line tools may be divided into the following classes:

T1 – generates no outputs which can directly or indirectly contribute to the executable code (including data) of the safety related system,

NOTE – T1 examples include: a text editor or a requirements or design support tool with no automatic code generation capabilities; configuration control tools.

T2 – supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software.

NOTE – T2 examples include: a test harness generator; a test coverage measurement tool; a static analysis tool.

T3 – generates outputs which can directly or indirectly contribute to the executable code of the safety related system.

NOTE – T3 examples include: a tool to change set-points during system operation; an optimizing compiler where the relationship between the source code program and the generated object code is not obvious; a compiler that incorporates an executable run-time package into the executable code.

# Test Environment Setup

*<Specify the Hardware, Software and any test data (pre defined) necessary for the test bed setup.>*

|  |  |  |  |
| --- | --- | --- | --- |
| Hardware / Software | Make /Model | Details | Remarks |
| Emulator / Simulator |  |  | *Refer note below* |
| Debugger |  |  |  |
| Interface Boards |  |  |  |
| Display Screen Panel |  |  |  |
| USB and Network Peripherals |  |  |  |
| Speaker |  |  |  |
| Flash Loader |  |  |  |
| RTOS |  |  |  |
| Desktop Machine |  |  |  |

*Note:*

* *For SIL3 software, a process simulator to simulate the Equipment Under Control should be used. Refer to C.5.18 of IEC61508-7 for an explanation of Process Simulator*
* *The emulator or simulator may need to be qualified. Emulator and Simulator should be included in list of Testing Tools in previous section on Testing tools*
* *The differences between the target computer and the emulator or simulator, and the effects of these differences on the ability to detect errors and verify functionality, should be indicated in this section*
* *Detection of those errors should be provided by other software verification process activities and specified in this section.*

## Test Equipments

|  |  |  |
| --- | --- | --- |
| Equipment | Equipment Make / Model No | Last Calibration date |
| Power Supply | AAA/cccc |  |
| Multimeter | Fluke /xxxx |  |
| Oscilloscope | HP /yyyyy |  |

Note: Calibration records of all test instruments to be maintained and enclosed with test results

# Defect Management

The defect management process that describe the procedures followed for collecting, analyzing and reporting the defects are explained below. It includes defect meeting, defect resolution, etc. Defect tools used – PMSmart is usually the tool used at HCL for defect Management.

## Responsibility

PM shall be owner/creator of Project in PMSmart (Defect Management Tool). PM shall be able to change the status of defects (see flowchart below).

## Defect Logging

Defect logging will be primarily done by Test engineers. However, others (developers, Tech leads, architects and PM) will also be able to log defects.

*<A brief summary about defect logging tool format and process that will be followed during test execution to be mentioned here>*

## Defect Analysis

Once the defect is logged, the PM is notified of defect by mail. PM analysis the defect to check if it’s a valid /invalid entry. If defect traces to an issue with specification, the PM shall take it up with Customer and resolve it accordingly.

For all valid entries, PM shall set the severity of defect – Major for all safety related defects and Major/Minor/Trivial for non safety defects and assign defect to developer. The developer completes the Test Incident report (or in other words the Impact Analysis) before starting to work on resolving the defect. Attached is copy of Test Incident report. Test Incident report is now incorporated into PMSmart.

## Defect Status Transition

<*Explain the flow of defect status and access role to change status. For example a defect initially when logged will be ‘Open’ state and get to ‘Assigned’ state when assigned to a developer , ‘Resolved’ when defect is corrected and sent back for re testing and finally ‘closed’ by the test resource after verification. Defect state can be changed to ‘Reopened’ if modified code did not resolve the defect*. >



## Defect Meetings

Detailed defect meeting are held with the development / defect fix team to rationalize and authenticate the reported defects.

*<Indicate the frequency of such meetings and who will be moderator of such meeting>*

## Defect Prevention Plan

This section is already covered in Functional safety management Plan (FSMP). The section heading is added to maintain completeness and to inform the reader to refer to FSMP

# Suspension and Resumption Requirements

Specify the criteria used to suspend all or portion of the testing activity on the test item. This situation arises in conditions like crashes and loss of major functionality during testing.

Specify the testing activities, which must be repeated, when testing is resumed. Following can be considered as guidelines

## Suspension

* Encountering showstopper at the application level or a show stopper at module level which is critical for the functioning of the application
* On transfer of wrong soft base
* On failure of test environment
* Any other circumstances or situation when the test team is convinced that the testing cannot be continued
* On request from <Customer>

## Resumption

* On resolving the situation, which resulted in the suspension or
* On specific request of the Customer on record mentioning the alleviating condition(s) allowing HCLT to proceed.

# Software Metrics

*<Indentify the Metrics to be collected during testing phase. Below is an example>*

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Data Source** | **Collection Frequency** |
| No of Test cases Executed | Test Summary Report | Daily |
| No of Major Defects Reported | Test Summary Report | Daily |
| No of Minor and Trivial Defects reported | Test Summary Report | Daily |
| Time Spent on testing | Test Log | Daily |
| KLOC (kilo lines of Code) | Test Summary Report | Before Testing of each release of Software |

# Traceability to IEC61508

## Mapping to Specifications

The Table below maps the Section 7.3, 7.5 and 7.7 of IEC61508-3, ed 2.0 2010 to this document.

| Req. ID | Specification description | Reference in this document |
| --- | --- | --- |
| **7.3 : Validation Plan for Software aspects of System Safety** | | |
| 7.3.2.1 | Planning shall be carried out to specify the steps, both procedural and technical, that will be used to demonstrate that the software satisfies its safety requirements. | This entire document covers this requirement |
| 7.3.2.2 | The validation plan for software aspects of system safety shall consider the following:  a) details of when the validation shall take place; | a – Validation entry criteria are defined in Section 4: Testing Methods. The FSMP shall have the Project schedule |
| b) details of those who shall carry out the validation; | Section2 : Project team |
| c) identification of the relevant modes of the EUC operation including:  1) preparation for use including setting and adjustment;  2) start up, teach, automatic, manual, semi-automatic, steady state operation;  3) re-setting, shut down, maintenance;  4) Reasonably foreseeable abnormal conditions and reasonably foreseeable operator misuse. | This will be part of test cases and not covered in test plan |
| d) identification of the safety-related software which needs to be validated for each mode of EUC operation before commissioning commences; | Section 2.3 : Test Scope |
| e) The technical strategy for the validation (for example analytical methods, statistical tests etc.); | Section 5 : Test Execution Strategy |
| f) in accordance with item e), the measures (techniques) and procedures that shall be used for confirming that each safety function conforms with the specified requirements for the  safety functions, and the specified requirements for software systematic capability;. | Section : 5.3 Testing techniques for requirements |
| g) the required environment in which the validation activities are to take place (for example, for tests this could include calibrated tools and equipment); | Section 9: Test Environment |
| h) the pass/fail criteria; | Section 4.4 Test Execution |
| i) the policies and procedures for evaluating the results of the validation, particularly failures | Section 10 : Defect Management |
| 7.3.2.3 | The validation shall give a rationale for the chosen strategy. The technical strategy for the validation of safety-related software shall include the following information:  a) choice of manual or automated techniques or both;  b) choice of static or dynamic techniques or both;  c) choice of analytical or statistical techniques or both.  d) choice of acceptance criteria based on objective factors or expert judgment or both. | Section 5.3 : Techniques for requirements |
| 7.3.2.4 | As part of the procedure for validating safety-related software aspects, the scope and contents of the validation plan for software aspects of system safety shall be agreed with the assessor or with a party representing the assessor, if required by the safety integrity level (see Clause 8 of IEC 61508-1). This procedure shall also make a statement concerning the presence of the assessor during testing. | Covered in Section 4.1 – Test Plan. And in Section 4.4 Test execution |
| 7.3.2.5 | The pass/fail criteria for accomplishing software validation shall include:  a) the required input signals with their sequences and their values;  b) the anticipated output signals with their sequences and their values; and | Pass /Fail Criteria for each test case shall be part of test case document. It shall describe the inputs and Outputs for each test case |
| c) other acceptance criteria, for example memory usage, timing and value tolerances. | Section 2.3.4 User Acceptance criteria |
| **7.5 :** **Programmable Electronics Integration** | | |
| 7.5.2.1 | Integration tests shall be specified during the design and development phase (see 7.4.3) to ensure the compatibility of the hardware and software in the safety-related programmable electronics. | Section 4.4 Test Execution |
| 7.5.2.2 | The software/PE integration test specification (hardware and software) shall state the following:  a) the split of the system into integration levels;  b) test cases and test data;  c) types of tests to be performed;  d) test environment including tools, support software and configuration description;  e) test criteria on which the completion of the test will be judged. | a - Section 5 : Test Strategy.  b- test cases will dealt separately  c- Section 5 : Test Strategy.  d- Section 9 :Test Environment  e- Section 4.4 :Test Execution |
| 7.5.2.3 | The software/PE integration test specification (hardware and software) shall distinguish between those activities which can be carried out by the developer on his premises and those that require access to the user's site. | Section 2.3 – Test Scope |
| 7.5.2.4 | The software/PE integration test specification (hardware and software) shall distinguish between the following activities:  a) merging of the software system on to the target programmable electronic hardware; | Section 4.4 –Test Case Execution |
| b) E/E/PE integration, i.e. adding interfaces such as sensors and actuators; | Section 4.4 –Test Case Execution |
| c) Applying the E/E/PE safety-related system to the EUC. | Section 4.4 –Test Case Execution |
| 7.5.2.5 | The software shall be integrated with the safety-related programmable electronic hardware in accordance with the software/PE integration test specification (hardware and software). | The Build Plan will describe the S/w-H/w integration. Refer Section 4.4 –Test Case Execution |
| 7.5.2.6 | During the integration testing of the safety-related programmable electronics (hardware and software), any change to the integrated system shall be subject to an impact analysis. The impact analysis shall determine all software modules impacted, and the  necessary re-verification activities. | Section 4.4 –Test Case Execution |
| 7.5.2.7 | Test cases and their expected results shall be documented for subsequent analysis. | Section 13 : Test Template |
| 7.5.2.8 | The integration testing of the safety-related programmable electronics (hardware and software) shall be documented, stating the test results, and whether the objectives and the test criteria have been met. If there is a failure, the reasons for the failure shall be documented. Any resulting modification or change to the software shall be subject to an impact analysis which shall determine all software elements/modules impacted, and the  necessary re-verification and re-design activities. | Test cases and Test result document covers this requirement (refer Section 13) .  Handling of failures are covered in Section 10 : Defect Management |
| **7.7 Software aspects of System Safety validation** | | |
| 7.7.2.1 | If the compliance with the requirements for safety-related software has already been established in the safety validation planning for the E/E/PE safety-related system (see 7.7 of IEC 61508-2), then the validation need not be repeated. | System Safety Validation will part of H/w-S/w Integration testing. |
| 7.7.2.2 | The validation activities shall be carried out as specified the in validation plan for software aspects of system safety. | Section 4 : Testing Methods |
| 7.7.2.3 | Depending on the nature of the software development, responsibility for conformance with 7.7 can rest with multiple parties. The division of responsibility shall be documented  during safety planning (see Clause 6 of IEC 61508-1). | Refer Section 8 :Project Team in Functional Safety Management plan |
| 7.7.2.4 | The results of validating the software aspects of system safety shall be documented. | Section 4.4 Test case Generation and Section 4.5 Test case execution |
| 7.7.2.5 | For each safety function, software safety validation shall document the following results:  a) a chronological record of the validation activities that will permit the sequence of activities to be retraced; | PM Smart (Defect management Tool) shall maintain all d effect history with date& time stamp |
| b) the version of the validation plan for software aspects of system safety (see 7.3) being used; | Revision History included at beginning of all documents. |
| c) the safety function being validated (by test or analysis), together with reference to the validation plan for software aspects of system safety; | Section 13 : Test Case template |
| d) tools and equipment used together with calibration data; | Section 9 : Test Environment Setup |
| e) the results of the validation activity;  f) Discrepancies between expected and actual results. | Section 13 : Test Case template |
| 7.7.2.6 | When discrepancies occur between expected and actual results, the analysis made and the decisions taken on whether to continue the validation, or to issue a change request and return to an earlier part of the development lifecycle, shall be documented as part of the results of validating the software aspects of system safety. | Section 4.4.1 Impact Analysis |
| 7.7.2.7 | The validation of safety-related software aspects of system safety shall meet the following requirements:  a) testing shall be the main validation method for software; analysis, animation and modeling may be used to supplement the validation activities; | Section4 : Testing Methods |
| b) the software shall be exercised by simulation of:  1) input signals present during normal operation;  2) anticipated occurrences;  3) undesired conditions requiring system action; | Section 5 : Test Execution Strategy |
| c) the supplier and/or developer (or the multiple parties responsible for compliance) shall make available the documented results of the validation of software aspects of system safety and all pertinent documentation to the system developer to enable his product to meet the requirements of IEC 61508-1 and IEC 61508-2. | Section 4.5 : Test Results |
| 7.7.2.8 | Software tools shall meet the requirements of 7.4.4. | Section 8 : Testing tools |
| 7.7.2.9 | The results of the validation of safety-related software aspects of system safety shall meet the following requirements:  a) the tests shall show that all of the specified requirements for safety-related software (see 7.2) are correctly met and the software does not perform unintended functions; | Section 4.4 Test Cases and Section 4.5 Test Results |
| b) test cases and their results shall be documented for subsequent analysis and independent assessment (see Clause 8 of IEC 61508-1) as required by the safety integrity level; | Section 13 : Test Case Template |
| c) the documented results of validating the software aspects of system safety shall state either (1) that the software has passed the validation or (2) the reasons for not passing the validation. | Section 4.5 Test Results |
| 7.9 Software Verification | | |
| 7.9.2.1 | The verification of software shall be planned (see 7.3) concurrently with the development, for each phase of the software safety lifecycle, and shall be documented. | Section 20.5: Verification Plan : FSMP001-Project Planning |
| 7.9.2.2 | The software verification planning shall refer to the criteria, techniques and tools to be used in the verification activities, and shall address: a) the evaluation of the safety integrity requirements; b) the selection and documentation of verification strategies, activities and techniques; c) the selection and utilisation of verification tools (test harness, special test software, input/output simulators etc.); d) the evaluation of verification results; e) the corrective actions to be taken. | a) Section 20: Quality Assurance Plan  b) Section 8.2.7 : Software Verification Process  c) Section 9.1 : Software Classification  d) Section 20.5 : Verification Plan:  e) Section 21.5 : Contrl of Non-conforming products  : FSMT001\_FunctionalSafetyManagementPlan |
| 7.9.2.3 | The software verification shall be performed as planned. NOTE Selection of techniques, measures for verification and the degree of independence of the verification activities will depend upon a number of factors and may be specified in application sector standards. The factors could include, for example: • size of project; • degree of complexity; • degree of novelty of design; • degree of novelty of technology. | Section 4.4 Test Execution of FSET006-Test Plan  FSET007-Test Case Template |
| 7.9.2.4 | Evidence shall be documented to show that the phase being verified has, in all respects, been satisfactorily completed. | 1.FSEF003-Test Incident Report 2.FSEF004-Test Summary Report |
| 7.9.2.5 | After each verification, the verification documentation shall include: a) identification of items to be verified; b) identification of the information against which the verification has been done; NOTE 1 Information against which the verification has been performed includes but is not limited to input from the previous lifecycle phase, design standards, coding standards and tools used. c) non-conformances. NOTE 2 Examples of non-conformances include software modules, data structures, and algorithms poorly adapted to the problem. | PSPF017-Walkthrough - Excel Format |
| 7.9.2.6 | All essential information from phase N of the software safety lifecycle needed for the correct execution of the next phase N+1 shall be available and shall be verified. Outputs from phase N include: a) adequacy of the specification, design, or code in phase N for: 1) functionality; 2) safety integrity, performance and other requirements of safety planning (see Clause 6); 3) readability by the development team; 4) testability for further verification; 5) safe modification to permit further evolution; b) adequacy of the validation planning and/or tests specified for phase N for specifying and describing the design of phase N; c) check for incompatibilities between: 1) the tests specified in phase N, and the tests specified in the previous phase N–1; 2) the outputs within phase N. | FSMT001\_FunctionalSafetyManagementPlan |
| 7.9.2.7 | Subject to the choice of software development lifecycle (see 7.1), the following verification activities shall be performed: a) verification of software safety requirements; b) verification of software architecture; c) verification of software system design; d) verification of software module design; e) verification of code; f) verification of data; g) verification of timing performance; h) software module testing (see 7.4.7); i) software integration testing (see 7.4.8); j) programmable electronics integration testing (see 7.5); k) software aspects of system safety validation (see 7.7). | FSMT001\_FunctionalSafetyManagementPlan |
| 7.9.2.8 | Verification of software safety requirements: after the software safety requirements specification has been completed, and before the next phase of software design and development begins, verification shall: a) consider whether the software safety requirements specification adequately fulfils the E/E/PE system safety requirements specification (see 7.10 of IEC 61508-1 and 7.2 of IEC 61508-2) for functionality, safety integrity, performance, and any other requirements of safety planning; b) consider whether the validation plan for software aspects of system safety adequately fulfils the software safety requirements specification; c) check for incompatibilities between: 1) the software safety requirements specification, and the E/E/PE system safety requirements specification (see 7.10 of IEC 61508-1 and 7.2 of IEC 61508-2); 2) the software safety requirements specification, and the validation plan for software aspects of system safety. | PSPF017-Walkthrough - Excel Format   FSET007-Test Case Template |
| 7.9.2.9 | Verification of software architecture: after the software architecture design has been completed, verification shall: a) consider whether the software architecture design adequately fulfils the software safety requirements specification; b) consider whether the integration tests specified in the software architecture design are adequate; c) consider whether the attributes of each major element/subsystem are adequate with reference to: 1) feasibility of the safety performance required; 2) testability for further verification; 3) readability by the development and verification team; 4) safe modification to permit further evolution. d) check for incompatibilities between the following: 1) the software architecture design, and the software safety requirements specification; 2) the software architecture design and its integration tests; 3) the software architecture design integration tests and the validation plan for software aspects of system safety. | PSPF017-Walkthrough - Excel Format   FSET007-Test Case Template |
| 7.9.2.10 | Verification of software system design: after the software system design has been completed, verification shall: a) consider whether the software system design (see 7.4.5) adequately fulfils the software architecture design; b) consider whether the specified tests of the software system integration (see 7.4.5) adequately fulfil the software system design (see 7.4.5); c) consider whether the attributes of each major element of the software system design specification (see 7.4.5) are adequate with reference to: 1) feasibility of the safety performance required; 2) testability for further verification; 3) readability by the development and verification team; 4) safe modification to permit further evolution. NOTE The software system integration tests may be specified as part of the software architecture integration tests. d) check for incompatibilities between: 1) the software system design specification (see 7.4.5), and the software architecture design; 2) the software system design specification (see 7.4.5), and the software system integration test specification (see.4.5); 3) the tests required by the software system integration test specification (see 7.4.5) and the software architecture integration test specification (see 7.4.3). | PSPF017-Walkthrough - Excel Format   FSET007-Test Case Template |
| 7.9.2.11 | Verification of software module design: after the design of each software module has been completed, verification shall: a) consider whether the software module design specification (see 7.4.5) adequately fulfils the software system design specification (see 7.4.5); b) consider whether the software module test specification (see 7.4.5) is adequate for the software module design specification (see 7.4.5); c) consider whether the attributes of each software module are adequate with reference to: 1) feasibility of the safety performance required (see software safety requirements specification); 2) testability for further verification; 3) readability by the development and verification team; 4) safe modification to permit further evolution. d) check for incompatibilities between: 1) the software module design specification (see 7.4.5), and the software system design specification (see 7.4.5); 2) (for each software module) the software module design specification (see 7.4.5), and the software module test specification (see 7.4.5); 3) the software module test specification (see 7.4.5), and the software system integration test specification (see 7.4.5). | PSPF017-Walkthrough - Excel Format   FSET007-Test Case Template |
| 7.9.2.12 | Verification of code: the source code shall be verified by static methods to ensure conformance to the software module design specification (see 7.4.5), the required coding standards (see 7.4.4), and the validation plan for software aspects of system safety. NOTE In the early phases of the software safety lifecycle, verification is static (for example inspection, review, formal proof, etc). Code verification includes such techniques as software inspections and walk-throughs. It is the combination of the results of code verification and software module testing that provides assurance that each software module satisfies its associated specification. From then onwards testing becomes the primary means of verification. | PSPF017-Walkthrough - Excel Format |
| 7.9.2.13 | Verification of data. a) The data structures shall be verified. b) The application data shall be verified for: 1) consistency with the data structures; 2) completeness against the application requirements; 3) compatibility with the underlying system software (for example, sequence of execution, run-time, etc.); and 4) correctness of the data values. c) All operational parameters shall be verified against the application requirements. d) All plant interfaces and associated software (i.e. sensors and actuators and off-line interfaces: see 7.2.2.12) shall be verified for: 1) detection of anticipated interface failures; 2) tolerance to anticipated interface failures. e) All communication interfaces and associated software shall be verified for an adequate level of: 1) failure detection; 2) protection against corruption; 3) data validation. | FSET007-Test Case Template |
| 7.9.2.14 | Verification of timing performance: predictability of behavior in the time domain shall be verified. | Section 2.3.4 : User Acceptance Criteria : FSET006-Test Plan : |

## Mapping to Testing Techniques in IEC61508-3 Annexure A & B

| Method | Technique | SIL1 | SIL2 | SIL3 | Remarks |
| --- | --- | --- | --- | --- | --- |
| **Probabilistic testing** |  | **--** | **R** | **R** | Not covered. Do not have experience in this technique |
| **Dynamic Analysis & testing** |  | **R** | **HR** | **HR** | ----- |
| Boundary value analysis | R | HR | HR | Covered in document . . |
| Error Guessing | R | R | R | Covered. |
| Error seeding | -- | R | R | Not covered. |
| Performance modeling | R | R | R | Not covered. |
| Equivalent classes and input partition testing | R | R | R | Covered. |
| Structured based Testing | HR | HR | HR | Covered. |
| **Data recording and analysis** |  | **HR** | **HR** | **HR** | Covered. |
| **Functional and Black box testing** |  | **HR** | **HR** | **HR** |  |
| Test case exec from cause consequence diagrams | -- | -- | R | Not covered. |
| Test case exec from Model based test cases | R | R | HR | Covered |
| Prototyping /animation | -- | -- | R | Not covered. |
| Equivalent classes and input partition testing, including boundary Value analysis | R | HR | HR | Covered |
| Process simulation | R | R | R | Not covered |
| **Performance Testing** |  | **R** | **R** | **HR** |  |
| Avalanche testing | R | R | HR | Covered |
| Response Timings and Memory constraints | HR | HR | HR | Covered |
| Performance requirements | HR | HR | HR | Covered |
| **Interface testing** |  | **R** | **R** | **HR** | Covered. |
| **Model based Testing** |  | **R** | **R** | **HR** | Covered. |
| **Test Management & automation tools** |  | **R** | **HR** | **HR** | Used |
| **Formal Verification** |  | **--** | **--** | **R** | Not covered. |
| **Modelling** |  | **R** | **R** | **HR** |  |
| Data Flow Diagrams | R | R | R | Covered |
| Finite State Machines | -- | R | HR | Covered as part of Model based testing |
| Formal Methods | -- | R | R | Not covered. |
| Time Petri Nets | -- | R | HR | Not covered. |
| Performance Modeling | R | HR | HR | Not covered. |
| Prototyping /animation | R | R | R | Not covered. |
| Structure diagrams | R | R | R | Not covered. |

1. In large Projects, Test Manager shall be Author of this document

   , [↑](#footnote-ref-1)