Test Plan

Version No.: 1.0

Date: 28-02-2019

Project Name: ADDI

Project Code:

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\*.

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version No | Date | Prepared by / Modified by | Significant Changes |
| 1.0 | 28-02-2019 | Khaja Hussain | Initial Draft |
|  |  |  |  |

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 |
| ADDI | IBM® Application Discovery and Delivery Intelligence (ADDI) |
| AD | IBM® Application Discovery (AD) |
| API | Application Program Interface |
| AWS | Amazon Web Services – Amazon Cloud |
| CCP | Hogan Condition Code Processing |
| CDMF | Hogan Common Data Management Facility |
| DDA | Hogan Demand Deposit Account |
| DG | Hogan Data Group (e.g. defined in CICS and matches a cobol copybook) |
| DI | IBM® Delivery Intelligence (DI) |
| ETL | Extract, Transform, Load |
| FPS | Hogan Function Processing System, used for Hogan online processing flows |
| HDB | Hogan Hierarchical Database (e.g. VSAM) |
| ODS | Hogan Online Delivery System |
| PAS | Hogan Platform Automation Support System |
| PCD | Hogan Process Control Data, allows setup and management of processing parameters via online screens |
| PD | Hogan Process Dictionary |
| PEM | Hogan Processing Environment Manager |
| SCM | Source Control Manger |
| SDB | Hogan Sequential Database |
| SPS | Hogan Scheduled Processing System, used for creating Hogan Reports |
| TDA | Hogan Time Deposit Account |
| UMB | Hogan Umbrella |

# Introduction

## Objective of this document

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

## Project Summary

IBM® Application Discovery and Delivery Intelligence (ADDI) is an analytical platform for application modernization. It uses cognitive technologies to analyze mainframe applications, so you can quickly discover and understand interdependencies and impacts of change.

There are two components in ADDI – Application Discovery (AD) and Delivery Intelligence (DI). While Delivery intelligence provides benefits of understanding applications run time behavior, Application Discovery part provides the enterprise view of applications landscape. The (DI) component is not in scope.

Using AD part of ADDI, a customer can view in graphical format, which represents the interrelationship between the components that make up the application. More information about IBM ADDI can be found at:

<https://www.ibm.com/support/knowledgecenter/en/SSRR9Q_5.0.4/IBM_AD_Analyze_User_Guide_OUT_KC/IBMADHighLevelArchitectureOverview.html>

DXC clients want to use IBM ADDI (AD) to investigate and learn about Hogan application. Hogan applications don’t directly utilize native z/OS constructs in Programs, but work through PEM. PEM provides abstraction layer between underlying z/OS and Hogan application. IBM ADDI or any other tool not provided by us cannot work.

Hogan Discovery API delivered through this project will help translate these native z/OS dependencies for Hogan. IBM ADDI will call Hogan Discovery API for translation.

## Test Scope

The initial phase will include all ‘must have’ requirements. These and any other requirements that get included must all be tested. At the end of Phase 1, a tester must be able to:

1. Create Automation test with as many steps as necessary
2. Retrieve it and have the ability to view it when running the test
3. Enter results and appropriate comments
4. View results

As the team works with the product they will define the needs for the second phase.

### Inclusions

|  |  |
| --- | --- |
| API | Test Description |
| Activity Controller | Activity Controller json file |
| Ccp Controller | Ccp Controller json file |
| Definition Group Controller | Definition Group Controller json file |
| Fps Controller | Fps Controller json file |
| Map Controller | Map Controller json file |
| Program Controller | Program Controller json file |
| Sdb Controller | Sdb Controller json file |
| Sim Controller | Sim Controller json file |
| Sps Controller | Sps Controller json file |
| Transaction Controller | Transaction Controller json file |
| Json Controller | Json Controller json file |

### Exclusions

**N/A**

### Test Assumptions

Test engineer responsible for testing the project. Then the Assumption is that all API unit test cases are executed and integrating testing also covered.

### User Acceptance Criteria

Identify 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.

The testing is considered to be successful provided

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

# Test Team

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

# Testing Strategy

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. | Test Plan checklist | Internal Approved Test Plan | Test Plan | PM, PM[[1]](#footnote-1) |

## Test case / Test Script Development

| Entry Criteria | Tasks | Verification | Exit Criteria | Deliverables | Document Owner ,Author |
| --- | --- | --- | --- | --- | --- |
| API Test Case Development | | | | | |
| Approved SRS | API /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 |

### Test Script

| 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 identified test cases | Approved Test scripts | Test Lead |

## 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 cases, Unit and integration testing. | Verification of test environment | Implementation of all checkpoints | Test Environment Setup Document / Automation | Test Lead |

## Test Execution

| Entry Criteria | Tasks | Verification | Release /Exit Criteria | Deliverables | Responsible Person |
| --- | --- | --- | --- | --- | --- |
| API Testing | | | | | |
| Approved API Test Case document  Testing Tools Calibration records | Execute Test cases | Automation Verification and validation of filled in Test Case | If All Unit test cases for the program are executed once and identified defects fixed | Test Report | Test Engineers |

### 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

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 identifies 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

| 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. |
| 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. |
| Trivial | Defects that affect limited areas of functionality that either can be work around or ignored. |

### Assessor’s Role during Test Execution

**N/A**

### Automation Test Case Execution

| 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 |

## Test Results

Test Results are published after the end of test case execution. This would 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 (API Testing)  Calibration records of test tools | Prepare Test Summary report | Test Result Review checklist | Completion of document | Test Report (API 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

* Automated 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 API testing and integration testing.

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

### API Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Generic Method | Specific Technique | SIL Level | | | Description |
| 1 | 2 | 3 |
| API and Integration testing: | Positive and Negative testing | HR | HR | HR | Entry & Exit point Testing.  Expected 100% coverage for each API |

### S/w Integration testing

N/A

### H/w – S/w Integration testing

N/A

## Additional Test Techniques

N/A

## Testing Techniques for Safety Requirements

N/A

# Code Coverage

N/A

## Coverage Criteria

N/A

# Test Automation

## Summary

* Reasons for doing Test Automation – like Customer mandated or many periodic releases envisaged or in general to improve internal resource productivity >
* Automating ADDI AP using Katalon Studio Tool
* 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 API

## API to be Test Automated

|  |  |
| --- | --- |
| Item to Test | Test Description |
| Activity Controller | Activity Controller json file |
| Ccp Controller | Ccp Controller json file |
| Definition Group Controller | Definition Group Controller json file |
| Fps Controller | Fps Controller json file |
| Map Controller | Map Controller json file |
| Program Controller | Program Controller json file |
| Sdb Controller | Sdb Controller json file |
| Sim Controller | Sim Controller json file |
| Sps Controller | Sps Controller json file |
| Transaction Controller | Transaction Controller json file |
| Json Controller | Json Controller json file |

# 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 suite 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classification | Tool and Version | Purpose | Criticality ( see note below) | Traceability to Tools Qualification Document |
| Testing Tools ( Automation testing) | 5.10.1 | Automation Testing | 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

|  |  |  |  |
| --- | --- | --- | --- |
| Hardware / Software | Make /Model | Details | Remarks |
| Laptops/Desktops | HP/Lenovo |  |  |
| Katalon Studio | 5.10.1 |  |  |

## 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.

## Responsibility

Excel sheet shall be owner/creator of Project. (Defect Management). Excel sheet shall be able to change the status of defects.

## 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.

## Defect Analysis

Once the defect is logged, the Excel sheet 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.

## Defect Status Transition

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 is held with the development / defect fix team to rationalize and authenticate the reported defects.

## 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 API 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

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Data Source** | **Collection Frequency** |
| 5 Test cases Executed | Test Summary Report | Daily |
| 1 Major Defects Reported | Test Summary Report | Daily |
| 3 Minor and Trivial Defects reported | Test Summary Report | Daily |

# Traceability to ADDI

## Mapping to Specifications

The Table below maps the Section 7.3, 7.5 and 7.7 of ADDI 2.0 2010 to this document.

| Req. ID | Specification description | Reference in this document |
| --- | --- | --- |
| **7.3 : Validation Plan for Software aspects of System Safety** | | |
| 7.3.2.2 | 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 have 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 API 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; | Excel sheet (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 : ADDI-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 utilization 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: Control of Non-conforming products  : ADDI\_Functional Safety Management Plan |
| 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 ADDI-Test Plan  ADDI-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.ADDI-Test Incident Report  2.ADDI-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. | ADDI-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. | ADDI\_ Functional Safety Management Plan |
| 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). | ADDI\_ Functional Safety Management Plan |
| 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. | ADDI-Walkthrough - Excel Format   ADDI-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. | ADDI-Walkthrough - Excel Format   ADDI-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). | ADDI-Walkthrough - Excel Format   ADDI-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). | ADDI-Walkthrough - Excel Format   ADDI-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. | ADDI-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. | ADDI-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 : ADDI-Test Plan : |

## Mapping to Testing Techniques in ADDI Annexure A & B

| Method | Technique | SIL1 | SIL2 | SIL3 | Remarks |
| --- | --- | --- | --- | --- | --- |
|  | Error Guessing | R | R | R | Covered. |
| Error Handling | -- | R | R | covered. |
| **API based Testing** |  | **R** | **R** | **HR** | Covered. |
| **Formal Verification and Validation** |  | **--** | **--** | **R** | covered. |

1. [↑](#footnote-ref-1)