Microsoft Office User

[Company name]  [Company address]

**Software Verification Plan of anti-lock braking system**

## . Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 10th September 17 | 1.0 | The software verification plan is created in MS-word | Fitash |
| 15 December 17 | 2.0 | Changes occurred during verification | Fitash |

Table of Contents

[. Revision History 2](#_Toc502882629)

[1. Scope and References 5](#_Toc502882630)

[1.1 Scope 5](#_Toc502882631)

[2. Definitions 5](#_Toc502882632)

[**Structural coverage:** 5](#_Toc502882633)

[**Statement coverage:** 5](#_Toc502882634)

[**Decision coverage:** 5](#_Toc502882635)

[**Condition coverage:** 5](#_Toc502882636)

[**MC/DC coverage:** 5](#_Toc502882637)

[**Safety-critical system:** 5](#_Toc502882638)

[**Harm:** 5](#_Toc502882639)

[**ASIL:** 6](#_Toc502882640)

[**Passenger car:** 6](#_Toc502882641)

[**Element:** 6](#_Toc502882642)

[2.3 Acronyms 6](#_Toc502882643)

[3. Software Verification and Validation Plan 6](#_Toc502882644)

[3.1 Purpose 6](#_Toc502882645)

[3.2 Definitions 6](#_Toc502882646)

[3.3 Verification and Validation Overview 6](#_Toc502882647)

[3.3.1 Organization 7](#_Toc502882648)

[3.3.2 Verification activities 7](#_Toc502882649)

[3.3.3 Tools, Techniques, and Methodologies 7](#_Toc502882650)

[3.4 Transition criteria 8](#_Toc502882651)

[3.4.1 Entry 8](#_Toc502882652)

[3.4.2 Internal 8](#_Toc502882653)

[3.4.3 Exit 8](#_Toc502882654)

[3.5 Reverification method 8](#_Toc502882655)

[3.6 Verification report 8](#_Toc502882656)

[Reference: 8](#_Toc502882657)

(This page is left blank intentionally)

# 1. Scope and References

## 1.1 Scope

ISO 26262 is standard for safety-related system that include electronic systems of automotive vehicles (passenger cars) having gross weight less than 3500 kilograms [1]. The scope of this plan is how the safety-related software of electronic system of automotive will be tested and verified with respect to ISO 26262. ISO 26262 divides the components of system on basis of criticality level ranging from A to D, A being the least critical and D being the most critical. ISO 26262 recommend to test these systems differently based on the criticality level. We are targeting software of ECU of anti-lock braking system which is of ASIL-D criticality as its failure can lead to loss of life. Its testing and verification activities are defined in this plan.

# 2. Definitions

The following terms, including those defined in other standards, are used as indicated in this standard.

### **Structural coverage:**

Structural coverage is that coverage which tells how much of source code is covered by the test cases it has different types including, statement coverage, decision coverage, mc/dc coverage etc.

### **Statement coverage:**

Statement coverage is a type of white box testing technique that says every line of code should be executed at least once.

### **Decision coverage:**

Decision coverage is a type of white box testing technique that says every decision of code should be executed with true and false at least once.

### **Condition coverage:**

Condition coverage is a type of white box testing technique that says every condition of code should be executed with true and false at least once.

### **MC/DC coverage:**

MC/DC coverage is a type of white box testing technique that says every condition, decision of code should be executed with true and false at least once and every entry and exit point of code should be invoked.

### **Safety-critical system:**

Safety critical systems are those systems whose failure can cause in loss of equipment or loss of life.

### **Harm:**

Physical injury or damage to health of person.

### **ASIL:**

ASIL is automotive safety integrity levels assigned to elements by safety engineers and testing of a component is based on ASIL level assigned to that element.

### **Passenger car:**

It is a vehicle designed and constructed primarily for movement of person and their luggage from one place to another without capacity of standing people.

### **Element:**

Part of system including component, hardware, software, hardware units and software units.

# 2.3 Acronyms

The following acronyms appear in this standard:

SDD Software Design Description   
SRS Software Requirements Specification   
ASIL Automotive safety integrity level   
ISO The international organization of standardization   
SQAP Software quality assurance plan

# 3. Software Verification and Validation Plan

This plan targets the software of Electronic control unit of anti-lock braking system which contains several modules containing speed calculator, fuel injector and controller. It is of ASIL-D software and should be tested accordingly

## 3.1 Purpose

This document explains the plan for verification of software for ECU of anti-lock braking system. ISO 26262 suggests to test the software according to requirements and structural coverage of code should be more than 80% depending upon the criticality level of software.

## 3.2 Definitions

This section shall describe the acronyms and notations used in the Plan. **Automotive safety integrity level** is assigned to each component of system by safety manager. Testing of any component is dependent upon its criticality level. The criticality levels are defined below:   
**ASIL A**: No injuries   
**ASIL B**: Light to moderate injuries   
**ASIL C**: Moderate to serious injuries   
**ASIL D**: Serious to life threatening injuries

## 3.3 Verification and Validation Overview

This section will explain the verification activities used to test the system.

### 3.3.1 Organization

Verification activities will be after every phase of life cycle; other than software all other activities will be verified by the checklists defined in SQA plan. As this software is of ASIL-D all review and verification activities should be by independent people, So four independent people from other teams will review conduct an audit of all activities and report to their project manager and project manager.

### 3.3.2 Verification activities

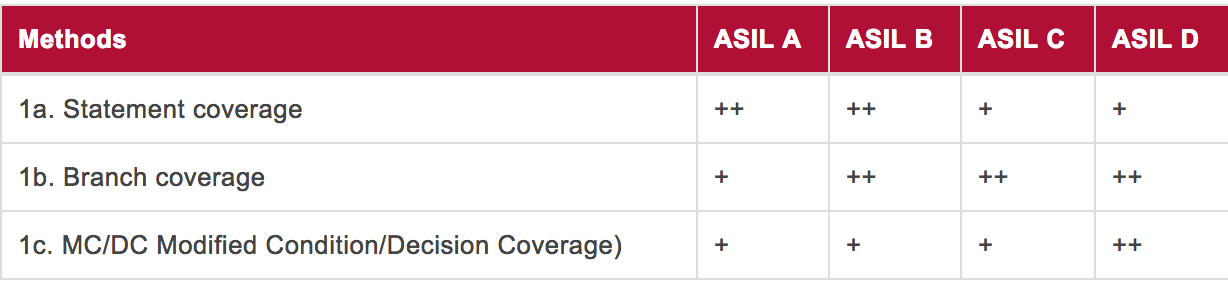
ISO 26262 suggests to test the system on basis of requirements and test cases should achieve more than 80% structural coverage depending upon the criticality level of the system. Software of ECU is of ASIL-D criticality level and ASIL-D suggest it that it should have 80% MC/DC of code as shown in table 1. Moreover, it should also have functional coverage and call coverage. 

Table 1

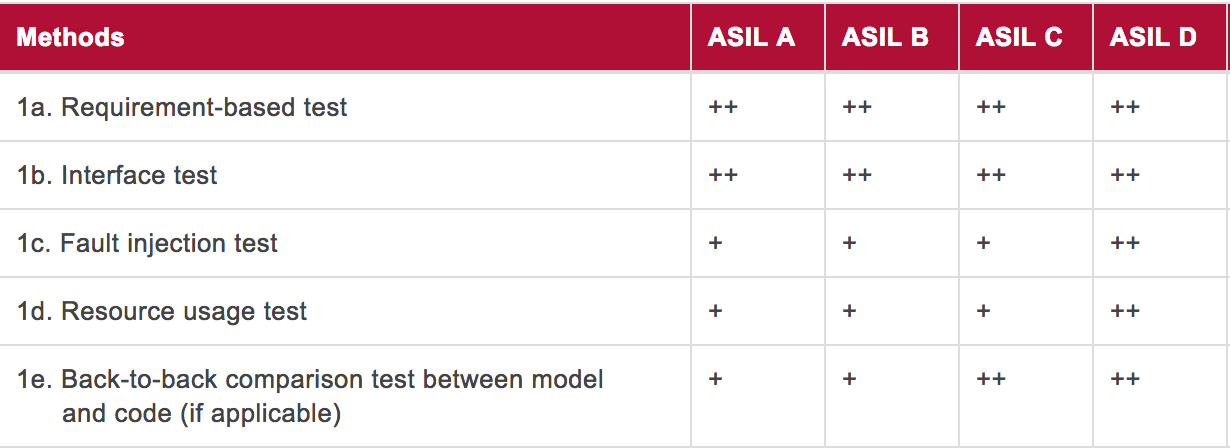
ISO 26262 suggest to test the software at unit level and as our software is of ASIL-D criticality the methods for testing at unit level is defined in table 2

Table 2

We will target and perform all kinds of testing defined in table 1 and table 2 as they are highly recommended for ASIL D.

### 3.3.3 Tools, Techniques, and Methodologies

We will use Verifysoft CTC++ tool to check MC/DC coverage of source code, test cases will be written manually. We will use the technique defined in literature to generate test cases from low level requirements but this effort will be manual. We will also use Lint tool to check misra compliance of our code.

# 3.4 Transition criteria

## 3.4.1 Entry

Reviews from SQAP will be done at the end of every activity and testing will start with the development.

## 3.4.2 Internal

Each developer will write unit test for the code he will write and independent tester will write test cases for requirement based testing.

## 3.4.3 Exit

This phase will exit after 90% of the reviews defined in SQAP are conducted and 80% of MC/DC is achieved.

# 3.5 Reverification method

Regression testing techniques will be used to retest the system and new test cases will be developed by independent tester as ASIL D requires independent testing.

# 3.6 Verification report

The testers and report auditor are independent and will report to project manager directly as it is required by ASIL D to have independent verification team. This report will contain test results from verifysoft and audit reports. These reports will then be forwarded to product owner and product owner will decide the next task to be done.

# 4. Deactivated Code

Deactivated code will be identified and proper rational will be provided and it will be made sure that deactivated code does not execute unless it is expected to through proper testing.

# 5. Dead Code

Dead code will be removed from the project.

## Reference:

[1] https://www.iso.org/standard/54591.html