Usage of tis-analyzer platform for ISO-26262

Version: 1.19

Contents

1. ISO-26262 Requirements

This document is about the verification of these requirements with tis-analyzer plateform or the use of tis-analyzer in order to achieve the verification of one requirement.

1.1. Analysis of boundary values

- Name: Analysis of boundary values
- Reference:
  - Table 11, Methods for deriving test cases for software unit testing
  - Table 14, Methods for deriving test cases for software integration testing
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It is possible to modelize a set of values or all the possible values of a given type. It will detect all possible undefined behavior of the software in the scope of tis-analyzer.

1.2. Analysis of requirements

- Name: Analysis of requirements
- Reference:
  - Table 11, Methods for deriving test cases for software unit testing
  - Table 14, Methods for deriving test cases for software integration testing
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  There are several sound technics implemented in tis-analyzer:
  - Abstract interpretation with the Value plugin
  - Weakest Precondition with the WP plugin

  All these plugins will check the requirements. As this is a very general rule, you can contact TrustInSoft for more informations for your personal case study.

1.3. Appropriate scheduling properties

- Name: Appropriate scheduling properties
- Reference:

  - Table 3, Principles for software architectural design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable.

1.4. Avoid global variables or else justify their usage

- Name: Avoid global variables or else justify their usage
- Reference:
  - Table 8, Design principles for software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	++	++

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks.

1.5. Back-to-back comparison test between model and code, if applicable

- Name: Back-to-back comparison test between model and code, if applicable
- Reference:
  - Table 10, Methods for software unit testing
  - Table 13, Methods for software integration testing
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	++	++

- Methodology:
  Not Applicable.

1.6. Branch coverage

- Name: Branch coverage
- Reference:
  - Table 12, Structural coverage metrics at the software unit level

- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It is possible to check the code coverage at the branch level.

1.7. Call coverage

- Name: Call coverage
- Reference:
  - Table 15, Structural coverage metrics at the software architectural level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It is possible to check the code coverage at the call level.

1.8. Control flow analysis

- Name: Control flow analysis
- Reference:
  - Table 4, Mechanisms for error detection at the software architectural level
  - Table 6, Methods for the verification of the software architectural design
  - Table 9, Methods for the verification of software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis

  - a sound analysis

  The value plugin analysis will detect all possible undefined behavior of the software in the scope of tis-analyzer. As this is a very general rule, you can contact TrustInSoft for more informations for your personal case study.

1.9. Correcting codes for data

- Name: Correcting codes for data
- Reference:
  - Table 5, Mechanisms for error handling at the software architectural level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	+

- Methodology:
  Not Applicable.

1.10. Data flow analysis

- Name: Data flow analysis
- Reference:
  - Table 6, Methods for the verification of the software architectural design
  - Table 9, Methods for the verification of software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  The value plugin analysis will detect all possible undefined behavior of the software in the scope of tis-analyzer. As this is a very general rule, you can contact TrustInSoft for more informations for your personal case study.

1.11. Detection of data errors

- Name: Detection of data errors
- Reference:
  - Table 4, Mechanisms for error detection at the software architectural level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	+

- Methodology:
  There are several sound technics implemented in tis-analyzer:
  - Abstract interpretation with the Value plugin
  - Weakest Precondition with the WP plugin

  All these plugins will detect data errors. As this is a very general rule, You can contact TrustInSoft for more informations for your personal case study.

1.12. Diverse software design

- Name: Diverse software design
- Reference:
  - Table 4, Mechanisms for error detection at the software architectural level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
o	o	+	++

- Methodology:
  Not Applicable.

1.13. Electronic control unit network environments

- Name: Electronic control unit network environments
- Reference:
  - Table 16, Test environments for conducting the software safety requirements verification
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable.

1.14. Enforcement of low complexity

- Name: Enforcement of low complexity
- Reference:
  - Table 1, Topics to be covered by modelling and coding guidelines
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable.

1.15. Enforcement of strong typing

- Name: Enforcement of strong typing
- Reference:
  - Table 1, Topics to be covered by modelling and coding guidelines
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable.

1.16. Error guessing

- Name: Error guessing
- Reference:
  - Table 11, Methods for deriving test cases for software unit testing
  - Table 14, Methods for deriving test cases for software integration testing
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	+

- Methodology:
  Not Applicable.

1.17. External monitoring facility

- Name: External monitoring facility
- Reference:
  - Table 4, Mechanisms for error detection at the software architectural level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
o	+	+	++

- Methodology:
  Not Applicable.

1.18. Fault injection test

- Name: Fault injection test
- Reference:
  - Table 10, Methods for software unit testing
  - Table 13, Methods for software integration testing
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It is possible to modelize a set of values or all the possible values of a given type. It will detect all possible undefined behavior of the software in the scope of tis-analyzer.

1.19. Formal notations

- Name: Formal notations
- Reference:
  - Table 2, Notations for software architectural design
  - Table 7, Notations for software unit design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	+

- Methodology:
  This is a very general rule. There are several formal methods that are implemented in tis-analyzer:
  - Abstract interpretation with the Value plugin
  - Weakest Precondition with the WP plugin

  You can contact TrustInSoft for more informations for your personal case study.

1.20. Formal verification

- Name: Formal verification
- Reference:
  - Table 6, Methods for the verification of the software architectural design
  - Table 9, Methods for the verification of software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
o	o	+	+

- Methodology:
  This is a very general rule. There are several formal methods that are implemented in tis-analyzer:
  - Abstract interpretation with the Value plugin
  - Weakest Precondition with the WP plugin

  You can contact TrustInSoft for more informations for your personal case study.

1.21. Function coverage

- Name: Function coverage
- Reference:
  - Table 15, Structural coverage metrics at the software architectural level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It is possible to check the code coverage at the function level.

1.22. Generation and analysis of equivalence classes

- Name: Generation and analysis of equivalence classes
- Reference:
  - Table 11, Methods for deriving test cases for software unit testing
  - Table 14, Methods for deriving test cases for software integration testing
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It is possible to modelize a set of values or all the possible values of a given type. It will detect all possible undefined behavior of the software in the scope of tis-analyzer.

1.23. Graceful degradation

- Name: Graceful degradation
- Reference:
  - Table 5, Mechanisms for error handling at the software architectural level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It is possible to modelize a set of values or all the possible values of a given type and check that the software has no undefined behavior while running only with critical functions.

1.24. Hardware-in-the-loop

- Name: Hardware-in-the-loop
- Reference:
  - Table 16, Test environments for conducting the software safety requirements verification
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	++	++

- Methodology:
  Not Applicable.

1.25. Hierarchical structure of software components

- Name: Hierarchical structure of software components
- Reference:
  - Table 3, Principles for software architectural design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable.

1.26. High cohesion within each software component

- Name: High cohesion within each software component
- Reference:
  - Table 3, Principles for software architectural design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  Not Applicable.

1.27. Independent parallel redundancy

- Name: Independent parallel redundancy
- Reference:
  - Table 5, Mechanisms for error handling at the software architectural level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
o	o	+	++

- Methodology:
  Not Applicable.

1.28. Informal notations

- Name: Informal notations
- Reference:
  - Table 2, Notations for software architectural design
  - Table 7, Notations for software unit design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	+	+

- Methodology:
  Not Applicable.

1.29. Initialization of variables

- Name: Initialization of variables
- Reference:
  - Table 8, Design principles for software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It will guaranty that there is no variables that are read before any initalization.

1.30. Inspection

- Name: Inspection
- Reference:
  - Table 9, Methods for the verification of software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  Not Applicable.

1.31. Inspection of the design

- Name: Inspection of the design
- Reference:
  - Table 6, Methods for the verification of the software architectural design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  Not Applicable.

1.32. Interface test

- Name: Interface test
- Reference:
  - Table 10, Methods for software unit testing
  - Table 13, Methods for software integration testing
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It will detect all possible undefined behavior of the software in the scope of tis-analyzer.

1.33. Limited use of pointers

- Name: Limited use of pointers
- Reference:
  - Table 8, Design principles for software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
o	+	+	++

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks.

1.34. MC/DC (Modified Condition/Decision Coverage)

- Name: MC/DC (Modified Condition/Decision Coverage)
- Reference:
  - Table 12, Structural coverage metrics at the software unit level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	++

- Methodology:
  Not Applicable.

1.35. Natural language

- Name: Natural language
- Reference:
  - Table 7, Notations for software unit design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable.

1.36. No dynamic objects or variables, or else online test during their creation

- Name: No dynamic objects or variables, or else online test during their creation
- Reference:
  - Table 8, Design principles for software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks.

1.37. No hidden data flow or control flow

- Name: No hidden data flow or control flow
- Reference:
  - Table 8, Design principles for software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  This is a very general rule. Check in this document for more precise rule to see if it is covered by tis-analyzer. In any case, you can contact TrustInSoft for more informations for your personal case study.

1.38. No implicit type conversions

- Name: No implicit type conversions
- Reference:
  - Table 8, Design principles for software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks. However, tis-analyzer can guaranty the absence of overflows.

1.39. No multiple use of variable names

- Name: No multiple use of variable names
- Reference:
  - Table 8, Design principles for software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks.

1.40. No recursions

- Name: No recursions
- Reference:
  - Table 8, Design principles for software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	++	++

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks.

1.41. No unconditional jumps

- Name: No unconditional jumps
- Reference:
  - Table 8, Design principles for software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks.

1.42. One entry and one exit point in subprograms and functions

- Name: One entry and one exit point in subprograms and functions
- Reference:
  - Table 8, Design principles for software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  The Inout plugin lists all kinds of inputs/outputs and can help to check the functions dependencies, and verify the entry point and exit point of a function.

1.43. Performing plausibility checks on calibration data

- Name: Performing plausibility checks on calibration data
- Reference:
  - Table C.1, Mechanisms for the detection of unintended changes of data
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable.

1.44. Plausibility check

- Name: Plausibility check
- Reference:
  - Table 4, Mechanisms for error detection at the software architectural level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	++

- Methodology:
  Differents strategies can be used inside tis-analyzer. A strong point of tis-analyzer is its capability to check annotations. By default, tis-analyzer can generate proof obligation that are automatically checked, and the user can add more specific annotations and see if they are automatically proved. You can contact TrustInSoft for more informations for your personal case study.

1.45. Prototype generation

- Name: Prototype generation
- Reference:
  - Table 6, Methods for the verification of the software architectural design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
o	o	+	++

- Methodology:
  Not Applicable.

1.46. Range checks of input and output data

- Name: Range checks of input and output data
- Reference:
  - Table 4, Mechanisms for error detection at the software architectural level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It is possible to modelize a set of values or all the possible values of a given type. It will detect all possible undefined behavior of the software in the scope of tis-analyzer.

1.47. Redundant storage of calibration data

- Name: Redundant storage of calibration data
- Reference:
  - Table C.1, Mechanisms for the detection of unintended changes of data
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	++

- Methodology:
  Not Applicable.

1.48. Requirements-based test

- Name: Requirements-based test
- Reference:
  - Table 10, Methods for software unit testing
  - Table 13, Methods for software integration testing
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  There are several sound technics implemented in tis-analyzer:
  - Abstract interpretation with the Value plugin
  - Weakest Precondition with the WP plugin

  All these plugins will check the requirements. As this is a very general rule, you can contact TrustInSoft for more informations for your personal case study.

1.49. Resource usage test

- Name: Resource usage test
- Reference:
  - Table 10, Methods for software unit testing
  - Table 13, Methods for software integration testing
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	++

- Methodology:
  Not Applicable.

1.50. Restricted coupling between software components

- Name: Restricted coupling between software components
- Reference:
  - Table 3, Principles for software architectural design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  Not Applicable.

1.51. Restricted size of interfaces

- Name: Restricted size of interfaces
- Reference:
  - Table 3, Principles for software architectural design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	+

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks.

1.52. Restricted size of software components

- Name: Restricted size of software components
- Reference:
  - Table 3, Principles for software architectural design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks.

1.53. Restricted use of interrupts

- Name: Restricted use of interrupts
- Reference:
  - Table 3, Principles for software architectural design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	++

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks.

1.54. Semantic code analysis

- Name: Semantic code analysis
- Reference:
  - Table 9, Methods for the verification of software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	+

- Methodology:
  The Value plugin of tis-analyzer implements abstract interpretation. The value plugin is:
  - a semantic analysis
  - a sound analysis

  It will detect all possible undefined behavior of the software in the scope of tis-analyzer.

1.55. Semi-formal notations

- Name: Semi-formal notations
- Reference:
  - Table 2, Notations for software architectural design
  - Table 7, Notations for software unit design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  This is a very general rule. There are several formal methods that are implemented in tis-analyzer:
  - Abstract interpretation with the Value plugin
  - Weakest Precondition with the WP plugin

  You can contact TrustInSoft for more informations for your personal case study.

1.56. Semi-formal verification

- Name: Semi-formal verification
- Reference:
  - Table 9, Methods for the verification of software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	++	++

- Methodology:
  This is a very general rule. There are several formal methods that are implemented in tis-analyzer:
  - Abstract interpretation with the Value plugin
  - Weakest Precondition with the WP plugin

  You can contact TrustInSoft for more informations for your personal case study.

1.57. Simulation of dynamic parts of the design

- Name: Simulation of dynamic parts of the design
- Reference:
  - Table 6, Methods for the verification of the software architectural design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	++

- Methodology:
  Not Applicable.

1.58. Statement coverage

- Name: Statement coverage
- Reference:
  - Table 12, Structural coverage metrics at the software unit level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	+	+

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It is possible to check the code coverage at the statement level.

1.59. Static code analysis

- Name: Static code analysis
- Reference:
  - Table 9, Methods for the verification of software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  The tis-analyzer platform is a static analysis platform that implements several differents technics to guaranty the absence of undefined behaviors. As this is a very general rule, you can contact TrustInSoft for more informations for your personal case study.

1.60. Static recovery mechanism

- Name: Static recovery mechanism
- Reference:
  - Table 5, Mechanisms for error handling at the software architectural level
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	+

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It is possible to modelize a set of values or all the possible values of a given type and check that for a wrong input, the software is able to fall back into a safe recovering mode.

1.61. Use of defensive implementation techniques

- Name: Use of defensive implementation techniques
- Reference:
  - Table 1, Topics to be covered by modelling and coding guidelines
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
o	+	++	++

- Methodology:
  The value plugin is:
  - a semantic analysis
  - a sound analysis

  It is possible to modelize a set of values or all the possible values of a given type and check that for a wrong input, the software is able to fall back into a safe mode.

1.62. Use of established design principles

- Name: Use of established design principles
- Reference:
  - Table 1, Topics to be covered by modelling and coding guidelines
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	++

- Methodology:
  Not Applicable.

1.63. Use of language subsets

- Name: Use of language subsets

- Reference:
  - Table 1, Topics to be covered by modelling and coding guidelines
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable. However, tis-analyzer works on C or C++ source code

1.64. Use of naming conventions

- Name: Use of naming conventions
- Reference:
  - Table 1, Topics to be covered by modelling and coding guidelines
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks.

1.65. Use of style guides

- Name: Use of style guides
- Reference:
  - Table 1, Topics to be covered by modelling and coding guidelines
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  Not Applicable. Tis-analyzer does not perform syntactic checks.

1.66. Use of unambiguous graphical representation

- Name: Use of unambiguous graphical representation
- Reference:
  - Table 1, Topics to be covered by modelling and coding guidelines

- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	++	++	++

- Methodology:
  Not Applicable.

1.67. Using error detecting codes

- Name: Using error detecting codes
- Reference:
  - Table C.1, Mechanisms for the detection of unintended changes of data
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
+	+	+	++

- Methodology:
  Not Applicable.

1.68. Vehicles

- Name: Vehicles
- Reference:
  - Table 16, Test environments for conducting the software safety requirements verification
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	++	++	++

- Methodology:
  Not Applicable.

1.69. Walkthrough

- Name: Walkthrough
- Reference:
  - Table 9, Methods for the verification of software unit design and implementation
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	+	o	o

- Methodology:
  Not Applicable.

1.70. Walkthrough of the design

- Name: Walkthrough of the design
- Reference:
  - Table 6, Methods for the verification of the software architectural design
- Description:

- Required ASIL:

ASIL A	ASIL B	ASIL C	ASIL D
++	+	o	o

- Methodology:
  Not Applicable.