ISO26262-6

This part of ISO 26262 specifies the requirements for product development at the software level for automotive applications, including the following:

**5 Initiation of product development at the software level**

**5.2 General**

The initiation of the *software development* is a planning activity, where *software development* sub-phases and their supporting processes (see ISO 26262-8 and ISO 26262-9) are **determined** and **planned** according to the extent and complexity of the item development.

The *software development* sub-phases and supporting processes are initiated by determining the appropriate methods in order to comply with the requirements and their respective ASIL.

The methods are supported by guidelines and tools, which are determined and planned for each sub-phase and supporting process.

NOTE Tools used for *software development* can include tools other than software tools.

EXAMPLE Tools used for testing phases.

The planning of the *software development* includes the coordination with the product development at the system level (see ISO 26262-4) and the hardware level (see ISO 26262-5).

**5.3.1 Prerequisites**

The following information shall be available:

 project plan (refined) in accordance with ISO 26262-4:2011, 5.5.1;

 safety plan (refined) in accordance with ISO 26262-4:2011, 5.5.2;

 technical safety concept in accordance with ISO 26262-4:2011, 7.5.1;

 system design specification in accordance with ISO 26262-4:2011, 7.5.2; and

 item integration and testing plan (refined) in accordance with ISO 26262-4:2011, 8.5.1.

**5.3.2 Further supporting information**

The following information can be considered:

 qualified software tools available (see ISO 26262-8:2011, Clause 11);

 qualified software components available (see ISO 26262-8:2011, Clause 12);

 design and coding guidelines for modelling and programming languages (from external source);

 guidelines for the application of methods (from external source); and

 guidelines for the application of tools (from external source).

**5.4 Requirements and recommendations**

|  |  |
| --- | --- |
| **Requirements (Not an option)** |  |
| **5.4.1** The activities and the determination of appropriate methods *for the product development at the software level* **shall be planned**. | Planned Activities:  Activity 1 with appropriated methods/tools:  Activity 2 with appropriated methods/tools:  … |
| **5.4.2** The tailoring of the lifecycle *for product development at the software level* **shall be performed** in accordance with ISO 26262-2:2011, 6.4.5, and based on the reference phase model given in Figure 2. | Tailoring of the lifecycle in accordance with ISO 26262-2:2011, 6.4.5 was Performed? |
| **5.4.3** If developing configurable software, Annex C **shall be applied**. | Is software configurable?  YES: Was Annex C Applied?YES/NO  NO: |
| **5.4.4** The *software development* process for the software of an item, including lifecycle phases, methods, languages and tools, **shall be consistent** across all the sub-phases of the software lifecycle and **shall be compatible** with the system and hardware development phases, such that the required data can be transformed correctly.  NOTE The sequencing of phases, tasks and activities, including iteration steps, for the software of an item is to ensure the consistency of the corresponding work products with the product development at the hardware level (see ISO 26262-5) and the product development at the system level (see ISO 26262-4). | Are the Methods, languages and tools consistent across all the sub-phases of the software lifecycle? YES/NO  **System and hardware development phase are compatible with the *software development* phase? (**such that the required data can be transformed correctly) YES/NO |
| **5.4.5** For each sub-phase of *software development*, the selection of the following, including guidelines for their application, **shall be carried out**:  a) Methods; and  b) Corresponding tools. | Sub Phase 1- Activities with appropriated methods/tools:  Sub Phase 2- Activities with appropriated methods/tools: |
| **5.4.6** The criteria that **shall be considered** when selecting a suitable modelling or programming language are:  a) an unambiguous definition;  EXAMPLE Syntax and semantics of the language.  b) the support for embedded real time software and runtime error handling; and  c) the support for modularity, abstraction and structured constructs.  Criteria that are not sufficiently addressed by the language itself shall be covered (enforced) by the corresponding **guidelines**, or by the **development environment**.  NOTE 1 The selected programming language (such as ADA, **C**, C++, Java, Assembler or a graphical modelling language) supports the topics given in 5.4.7.  **Programming or modelling guidelines** can be used to comply with these topics.  NOTE 2 Assembly languages can be used for those parts of the software where the use of high-level programming languages is not appropriate, such as low-level software with interfaces to the hardware, interrupt handlers, or time-critical algorithms. | **The criteria considered for the selected modelling or programming language are:**  **a)** an unambiguous definition;  **b)** the support for embedded real time software and runtime error handling; and  **c)** the support for modularity, abstraction and structured constructs.  **Selected Language:** ADA, C, C++, Java, Assembler, graphical modelling language |
| **5.4.7** To support the correctness of the design and implementation, the design and coding guidelines for the modelling, or programming languages, shall address the topics listed in Table 1.  NOTE 1 Coding guidelines are usually different for different programming languages.  NOTE 2 Coding guidelines can be different for model-based development.  NOTE 3 Existing coding guidelines can be modified for a specific item development.  EXAMPLE MISRA C[3] and MISRA AC AGC[4] are coding guidelines for the programming language C. |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics** | | ASIL | | | |
| A | B | C | D |
| 1a | Enforcement of low complexity a | ++ | ++ | ++ | ++ |
| 1b | Use of language subsets b | ++ | ++ | ++ | ++ |
| 1c | Enforcement of strong typing c | ++ | ++ | ++ | ++ |
| 1d | Use of defensive implementation techniques | o | + | ++ | ++ |
| 1e | Use of established design principles | + | + | + | ++ |
| 1f | Use of unambiguous graphical representation | + | ++ | ++ | ++ |
| 1g | Use of style guides | + | ++ | ++ | ++ |
| 1h | Use of naming conventions | ++ | ++ | ++ | ++ |

a An appropriate compromise of this topic with other methods in this part of ISO 26262 may be required.

b The objectives of method 1b are

Exclusion of ambiguously defined language constructs which may be interpreted differently by different modellers, programmers, code generators or compilers.

Exclusion of language constructs which from experience easily lead to mistakes, for example assignments in conditions or identical naming of local and global variables.

Exclusion of language constructs which could result in unhandled run-time errors.

c The objective of method 1c is to impose principles of strong typing where these are not inherent in the language.

**6 Specification of software safety requirements**

|  |  |
| --- | --- |
| Objectives |  |
| The first objective of this sub phase is to specify the software safety requirements (Requirements related to software) They are derived from the technical safety concept and the system design specification. | Are the software safety requirements specified based on the Technical safety concept and the system design specification? |
| The second objective is to detail the hardware-software interface requirements initiated in ISO 26262-4:2011,  Clause 7. | Are the hardware-software interface requirements detailed? |
| The third objective is to verify that the software safety requirements and the hardware-software interface requirements are consistent with the technical safety concept and the system design specification. | Are the software safety requirements and the hardware-software interface requirements consistent with the technical safety concept and the system design specification? |

**6.3 Inputs to this clause**

**6.3.1 Prerequisites**

The following information shall be available:

 technical safety concept in accordance with ISO 26262-4:2011, 7.5.1;

 system design specification in accordance with ISO 26262-4:2011, 7.5.2;

 hardware-software interface specification in accordance with ISO 26262-4:2011, 7.5.3;

 safety plan (refined) in accordance with 5.5.1;

 software verification plan in accordance with 5.5.2.

**6.3.2 Further supporting information**

The following information can be considered:

 hardware design specification (see ISO 26262-5:2011, 7.5.1);

 guidelines for the application of methods (from external source).

**6.4 Requirements and recommendations**

|  |  |
| --- | --- |
| **6.4.1** The software safety requirements **shall address** each software-based function whose failure could lead to a violation of a technical safety requirement allocated to software.  EXAMPLE Functions whose failure could lead to a violation of a safety requirement can be:  - functions that enable the system to achieve or maintain a safe state;  - functions related to the detection, indication and handling of faults of safety-related hardware elements;  - functions related to the detection, notification and mitigation of faults in the software itself;  NOTE 1 These include both the self-monitoring of the software in the operating system and application-specific self-monitoring of the software to detect, indicate and handle systematic faults in the application software.  - functions related to on-board and off-board tests;  NOTE 2 On-board tests can be carried out by the system itself or through other systems within the vehicle network during operation and during the pre-run and post-run phase of the vehicle.  NOTE 3 Off-board tests refer to the testing of the safety-related functions or properties during production or in service.  - functions that allow modifications of the software during production and service; and  - Functions related to performance or time-critical operations. | Arefunctions that enable the system to achieve or maintain a safe state addressed?  Are functions related to the detection, indication and handling of faults of safety-related hardware elements addressed?  Are functions related to the detection, notification and mitigation of faults in the software itself?  Are functions that allow modifications of the software during production and service addressed?  Are functions related to performance or time-critical operations addressed?  Shall produce a Work Product **Software safety requirements specification** |
| 6.4.2 The specification of the software safety requirements shall be derived from the technical safety concept and the system design in accordance with ISO 26262-4:2011, 7.4.1 and 7.4.5, and shall consider:  a) the specification and management of safety requirements in accordance with ISO 26262-8:2011,Clause 6;  b) the specified system and hardware configurations;  EXAMPLE 1 Configuration parameters can include gain control, band pass frequency and clock prescaler.  c) the hardware-software interface specification;  d) the relevant requirements of the hardware design specification;  e) the timing constraints;  EXAMPLE 2 Execution or reaction time derived from the required response time at the system level.  f) the external interfaces; and  EXAMPLE 3 Communication and user interfaces.  g) each operating mode of the vehicle, the system, or the hardware, having an impact on the software.  EXAMPLE 4 Operating modes of hardware devices can include default, initialization, test, and advanced modes. | Was the specification of the Software Safety Requirements derived from the Technical Safety Concept and from the System Design Specification? |
| **6.4.3** If ASIL decomposition is applied to the software safety requirements, ISO 26262-9:2011, Clause 5,  shall be complied with. | Shall produce a Work Product **Software safety requirements specification** |
| **6.4.4** The hardware-software interface specification initiated in ISO 26262-4:2011, Clause 7, **shall be** detailed down to a level allowing the correct control and usage of hardware, and shall describe each safety-related dependency between hardware and software. | Shall produce a Work Product **Hardware-software interface specification (refined)** |
| **6.4.5** If other functions in addition to those functions for which safety requirements are specified in 6.4.1 are  carried out by the embedded software, these functions **shall be specified**, or else a reference made to their  specification. | Shall produce a Work Product **Software safety requirements specification** |
| **6.4.6** The verification of the software safety requirements and the verification of the refined specification of the hardware software interface shall be planned in accordance with ISO 26262-8:2011, Clause 9. | Shall produce a Work Product **Software verification plan (refined)** |
| **6.4.7** The refined hardware-software interface specification shall be verified jointly by the persons responsible for the system, hardware and software development. | Shall produce a Work Product **Software verification report** |
| **6.4.8** The software safety requirements and the refined hardware-software interface requirements shall be  verified in accordance with ISO 26262-8:2011, Clauses 6 and 9, to show their:  a) compliance and consistency with the technical safety requirements;  b) compliance with the system design; and  c) consistency with the hardware-software interface. | Shall produce a Work Product **Software verification report** |
|  |  |
|  |  |
|  |  |

**6.5 Work products**

**6.5.1 Software safety requirements specification** resulting from requirements 6.4.1 to 6.4.3 and 6.4.5.

**6.5.2 Hardware-software interface specification (refined)** resulting from requirement 6.4.4.

NOTE This work product refers to the same work product as given in ISO 26262-5:2011 6.5.2

**6.5.3 Software verification plan (refined)** resulting from requirement 6.4.6.

**6.5.4 Software verification report** resulting from requirements 6.4.7 and 6.4.8.

**7 Software architectural design**

The software architectural design represents all software components and their interactions in a hierarchical structure. Static aspects, such as interfaces and data paths between all software components, as well as dynamic aspects, such as process sequences and timing behavior are described.

NOTE The software architectural design is not necessarily limited to one microcontroller or ECU, and is related to the technical safety concept and system design. The software architecture for each microcontroller is also addressed by this chapter.

In order to develop a software architectural design both software safety requirements as well as all non-safety related requirements are implemented. Hence in this sub-phase safety-related and non-safety-related requirements are handled within one development process.

The software architectural design provides the means to implement the software safety requirements and to manage the complexity of the software development.

**7.1 Objectives**

|  |  |
| --- | --- |
| The first objective of this sub phase is to develop a software architectural design that realizes the software safety requirements. |  |
| The second objective of this sub phase is to verify the software architectural design. |  |

**7.4 Requirements and recommendations**

|  |  |  |
| --- | --- | --- |
|  |  | Comment/Answer |
| **7.4.1** To ensure that the software architectural design captures the information necessary to allow the subsequent development activities to be performed correctly and effectively, the software architectural design **shall be described** with appropriate levels of abstraction by using the notations for software architectural design listed in Table 2. | Notation used to describe the software architectural design: |  |
| Informal notations |  |
| Semi-formal notations |  |
| Formal notations |  |
| **7.4.2** During the development of the software architectural design the following **shall be considered**:  a) the verifiability of the software architectural design;  NOTE This implies bi-directional traceability between the software architectural design and the software safety requirements.  b) the suitability for configurable software;  c) the feasibility for the design and implementation of the software units;  d) the testability of the software architecture during software integration testing; and  e) the maintainability of the software architectural design. | a) Was traceability between the software architectural design and the software safety requirements used? |  |
| b) If configurable software is used, was the software architectural design suit? |  |
| c) Is the design and implementation of the software units feasible? |  |
| d) Can the software architecture during software integration testing be tested? |  |
| e) Is software architectural design maintainable? |  |
| **7.4.3** In order to avoid failures resulting from high complexity, the software architectural design **shall** exhibit the following properties by use of the principles listed in Table 3:  a) modularity;  b) encapsulation; and  c) simplicity.  NOTE An appropriate compromise between the methods listed in Table 3 can be necessary since the methods are not mutually exclusive. | Does the software architectural design exhibit |  |
| a) modularity; |  |
| b) encapsulation; and |  |
| c) simplicity? |  |
|  |  |
| **7.4.4** The software architectural design **shall be developed** down to the level where all software units are identified. | Are all software units identified? |  |
| **7.4.5** The software architectural design **shall describe**:  a) the static design aspects of the software components; and  NOTE 1 Static design aspects address:  - the software structure including its hierarchical levels;  - the logical sequence of data processing;  - the data types and their characteristics;  - the external interfaces of the software components;  - the external interfaces of the software; and  - the constraints including the scope of the architecture and external dependencies.  NOTE 2 In the case of model-based development, modelling the structure is an inherent part of the overall modelling activities.  b) the dynamic design aspects of the software components.  NOTE 1 Dynamic design aspects address:   the functionality and behaviour;   the control flow and concurrency of processes;   the data flow between the software components;   the data flow at external interfaces; and   the temporal constraints.  NOTE 2 To determine the dynamic behaviour (e.g. of tasks, time slices and interrupts) the different operating states (e.g. power-up, shut-down, normal operation, calibration and diagnosis) are considered.  NOTE 3 To describe the dynamic behaviour (e.g. of tasks, time slices and interrupts) the communication relationships and their allocation to the system hardware (e.g. CPU and communication channels) are specified. | In the Static design, Is the software structure including its hierarchical levels described? |  |
| In the Static design, Is the logical sequence of data processing described? |  |
| In the Static design, Is the data types and their characteristics described? |  |
| In the Static design, Is the external interfaces of the software components described? |  |
| In the Static design, Is the external interfaces of the software described? |  |
| In the Static design, Is the constraints including the scope of the architecture and external dependencies described? |  |
| In the Dynamic design, is the functionality and behavior described? |  |
| In the Dynamic design, is the control flow and concurrency of processes described? |  |
| In the Dynamic design, is the data flow between the software components described? |  |
| In the Dynamic design, is the data flow at external interfaces described? |  |
| In the Dynamic design, is the temporal constraints described? |  |
| **7.4.6** Every safety-related software component shall be categorized as one of the following:  a) newly developed;  b) reused with modifications; or  c) reused without modifications. | Are all safety related software component categorized as |  |
| a) newly developed; |  |
| b) reused with modifications; |  |
| c) reused without modifications. |  |
| **7.4.7** Safety-related software components that are newly developed or reused with modifications shall be developed in accordance with ISO 26262.  NOTE In these cases ISO 26262-8:2011, Clause 12, does not apply. | Are all safety related software component newly developed or reused with modifications developed in accordance with ISO 26262? |  |
| **7.4.8** Safety-related software components that are reused without modifications shall be qualified in  accordance with ISO 26262-8:2011, Clause 12. | Are all safety related software component reused without modifications qualified in accordance with ISO 26262-8:2011, Clause 12? |  |
| **7.4.9** The software safety requirements shall be allocated to the software components. As a result, each software component **shall be developed** in compliance with the highest ASIL of any of the requirements allocated to it.  NOTE Following this allocation, further refinement of the software safety requirements can be necessary. | Are The software safety requirements allocated to the software components? |  |
| Are all software components developed in compliance with the highest ASIL of the requirements allocated to it? |  |
| **7.4.10** If the embedded software has to implement software components of different ASILs, or safety-related  and non-safety-related software components, then all of the embedded software **shall be treated** in  accordance with the highest ASIL, unless the software components meet the criteria for coexistence in  accordance with ISO 26262-9:2011, Clause 6. | Is the embedded software treated in accordance with the highest ASIL? |  |
| **7.4.11** If software partitioning (see Annex D) is used to implement freedom from interference between software components it shall be ensured that:  a) the shared resources are used in such a way that freedom from interference of software partitions is ensured;  NOTE 1 Tasks within a software partition are not free from interference among each other.  NOTE 2 One software partition cannot change the code or data of another software partition nor command non shared resources of other software partitions.  NOTE 3 The service received from shared resources by one software partition cannot be affected by another software partition. This includes the performance of the resources concerned, as well as the rate, latency, jitter and duration of scheduled access to the resource.  b) the software partitioning is supported by dedicated hardware features or equivalent means (this requirement applies to ASIL D, in accordance with 4.3);  c) the part of the software that implements the software partitioning is developed in compliance with the  same or an ASIL higher than the highest ASIL assigned to the requirements of the software partitions;  and NOTE In general the operating system provides or supports software partitioning.  d) the verification of the software partitioning during software integration and testing (in accordance with  Clause 10) is performed. | Are shared resources used in such a way that freedom from interference of software partitions is ensured? |  |
| is the software partitioning supported by dedicated hardware features or equivalent means (this requirement applies to ASIL D, in accordance with 4.3); |  |
| Is the part of the software that implements the software partitioning developed in compliance with the same or an ASIL higher than the highest ASIL assigned to the requirements of the software partitions? |  |
| Was the verification of the software partitioning during software integration and testing (in accordance with Clause 10) performed? |  |
| **7.4.12** An analysis of dependent failures in accordance with ISO 26262-9:2011, Clause 7, **shall be carried out** if the implementation of software safety requirements relies on freedom from interference or sufficient independence between software components. | if the implementation of software safety requirements relies on freedom from interference or sufficient independence between software components, was an analysis of dependent failures in accordance with ISO 26262-9:2011, Clause 7, **carried out**? |  |
| **7.4.13** Safety analysis **shall be carried out** at the software architectural level in accordance with ISO 26262-9:2011, Clause 8, in order to:   identify or confirm the safety-related parts of the software; and   support the specification and verify the efficiency of the safety mechanisms.  NOTE Safety mechanisms can be specified to cover both issues associated with random hardware failures as well as software faults. | Was a Safety analysis **carried out** at the software architectural level in accordance with ISO 26262-9:2011, Clause 8, in order: |  |
| To identify or confirm the safety-related parts of the software? |  |
| To support the specification and verify the efficiency of the safety mechanisms. |  |
| **7.4.14** To specify the necessary software safety mechanisms at the software architectural level, based on  the results of the safety analysis in accordance with 7.4.13, mechanisms for error detection as listed in Table 4 shall be applied.  NOTE When not directly required by technical safety requirements allocated to software, the use of software safety mechanisms is reviewed at the system level to analyze the potential impact on the system behavior. |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |