



ASAM

Association for Standardization of
Automation and Measuring Systems

ASAM SCDL

Safety Concept Description Language

Part 1 of 3

Notation Specification

Version 1.6.0

Date: 2021-11-09

Base Standard

Disclaimer

This document is the copyrighted property of ASAM e.V.
Any use is limited to the scope described in the license terms. The license
terms can be viewed at www.asam.net/license

Table of Contents

1	Foreword	6
2	Introduction	7
	2.1 Overview	7
	2.2 Motivation.....	7
	2.3 Scope.....	7
	2.4 Structure of specification.....	8
3	Relations to Other standards	9
	3.1 Relation to Earlier Releases.....	9
	3.2 Relationship with ISO 26262	9
4	Terms and Definitions	10
	4.1 Assignment	10
	4.2 Element	10
	4.3 Freedom from interference	10
	4.4 Functional requirement.....	10
	4.5 Functional safety	10
	4.6 Independence requirement	10
	4.7 Intended functionality.....	10
	4.8 Non-functional requirement.....	10
	4.9 Other technology	11
	4.10 Safety architecture	11
	4.11 Safety mechanism	11
	4.12 Safety requirement	11
	4.13 Weighting	11
	4.14 Abbreviations.....	11
5	About SCDL	12
	5.1 Overview	12
	5.2 Introduction and concept.....	12
	5.3 Scope.....	14
	5.4 Basics of the SCDL notation.....	14
6	Basic definition of SCDL	15
	6.1 Overview	15

6.2 Definition of symbols	15
6.2.1 Notation of requirements.....	18
6.2.2 Notation of interactions	19
6.2.3 Notation for system boundary interactions	20
6.2.4 Prohibitions and exceptions for interaction notation.....	24
6.2.5 Notation for elements	25
6.2.6 Constraints.....	27
6.2.7 Connecting lines	28
6.3 Meaning of combinations of symbols	29
6.3.1 Allocation of requirements to elements.....	29
6.3.2 How to make requirement groups	31
6.3.3 How to describe constraints by using connecting lines from the pairing line between requirement groups.....	38
6.3.4 Detailing constraints between requirement groups	41
6.3.5 Notation for freedom from interference.....	41
6.3.6 Branching of an interaction line between requirements	43
 7 Bibliography	 45
 Appendix: A. SCDL metamodel	 46
A.1. Overview	46
A.2. Scope	48
A.3. SCDLType.....	48
A.3.1. Specializations	48
A.3.2. Attributes.....	48
A.3.3. WeightableType	48
A.3.4. Generalizations	48
A.3.5. Specializations	48
A.3.6. Association Ends	48
A.4. Weighting.....	49
A.5. AbstractRequirement.....	49
A.5.1. Generalizations	49
A.5.2. Constraints.....	49
A.5.3. Specializations	49
A.5.4. Attributes.....	49
A.5.5. Association Ends	49
A.6. Element.....	49
A.6.1. Generalizations	49
A.6.2. Constraints.....	49
A.6.3. Association Ends	49
A.7. Requirement	50
A.7.1. Generalizations	50
A.7.2. Association Ends	50
A.8. Constraint	50
A.8.1. Generalizations	50
A.8.2. Association Ends	50
A.9. ConstraintPairing (Relation with constraints).....	50
A.9.1. Generalizations	50

A.9.2. Association Ends	50
A.10. Interaction.....	51
A.10.1.Generalizations	51
A.10.2.Constraints.....	51
A.10.3.Association Ends	51
A.11. RequirementGroup.....	51
A.11.1.Generalizations	51
A.11.2.Constraints.....	51
A.11.3.Association Ends	51
A.12. ConstraintTarget (Target to which constraints are applied)	51
A.12.1.Generalizations	51
A.12.2.Specializations	51
A.12.3.Association Ends	52
A.13. IndependencyTarget	52
A.13.1.Generalizations	52
A.13.2.Specializations	52
A.14. CoexistenceTarget	52
A.14.1.Generalizations	52
A.14.2.Association Ends	52
A.15. InterferenceTarget.....	52
A.15.1.Specializations	52
A.16. RequirementGroupPairing.....	52
A.16.1.Generalizations	52
A.16.2.Constraints.....	53
A.16.3.Association Ends	53
A.17. RequirementPairing (Partial pairing)	53
A.17.1.Generalizations	53
A.17.2.Constraints.....	53
A.17.3.Association Ends	53
A.18. Relationships between SCDL metamodel and figures	53
A.18.1.Relationships in figures in the specification	53
A.18.2.Relationships considering figures used for use case examples	55

1 Foreword

SCDL (Safety Concept Description Language) is a semi-formal notation to describe ISO 26262 safety architectures, namely safety concepts. This includes safety requirement specifications, element architectures, requirements allocation on elements, ASIL assignments, decompositions for safety mechanisms and others. SCDL as a vendor-independent language targets modeling methods for ISO 26262 by providing intuitive graphical representations and straightforward processes. Tools based on SCDL support the development, design, analysis, and verification of ISO 26262 artefacts. Interoperability and exchangeability of methods and artefacts are provided.

First, introduction to SCDL is described in this document. Terms and definitions for the standard follow after the introduction. It is especially intended that terms have the same meaning as in ISO 26262. Finally, notation of SCDL is defined. Thus, all symbols for SCDL are standardized in this document.

2 Introduction

2.1 Overview

ISO 26262 requires a top-down approach for safety explanation, and the safety concept is a key deliverable. SCDL (Safety Concept Description Language) provides a visualization method for intuitively explaining safety concepts. The SCDL can satisfy the requirements of ISO 26262 as a semi-formal notation.

2.2 Motivation

Nowadays, it is strongly recommended to show product safety with clear evidence for various products ranging from large-scale systems supporting social infrastructure to consumer products. The trend has been seen remarkably through safety activities in the field of industrial facilities and equipment such as chemical plants, nuclear power plants, and machine tools according to the publication of a series of ISO/IEC safety standards, visualization of safety design of the systems, and efforts to enhance accountability to the third party. In the automotive industry, such safety activities have been strengthened since ISO 26262¹ was published.

To visualize safety design, safety-related activities for the system such as developing safety concepts, designing for safety, and verifying safety are performed and explained, i.e. how the safety of the system is incorporated in the design or how safety is ensured needs to be explained clearly.

Additionally, safety is an important theme to ensure maintainability or to discuss security measures for the system.

Primary factors required for safety activities such as safety concept, safety design, and safety verification are requirements and architecture. In such safety activities, a notation to visually express the relationship between the requirements and the architecture as well as a methodology to verify it is needed. Responding to such needs, this document provides the notation method; Safety Concept Description Language (SCDL) which is useful for design engineers/verification engineers involved in product development and the third parties (evaluators) involved in product certification.

2.3 Scope

This document defines the grammar of SCDL and provides application guidance. SCDL is a notation method to organize and describe the safety design of the system from the architecture point of view. It is possible to specify safety design using existing notation methods, however, a notation method, which provides continuous support for consideration or argumentation of the safety design from the perspective of architecture consistently, is easy to understand, and allows review activities focusing on the safety design, is expected.

To address these issues, SCDL deals with architecture as the main topic for safety design and hierarchically expresses identification of safety mechanisms and safety-related portions.

¹ISO 26262 : 2018. Road Vehicles -- Functional Safety. ISO Standard.

In an architecture designed using SCDL, requirements² and system constituents³ to which requirements are allocated are distinguished. Also, dependencies between the requirements, grouping of the requirements, requirements which become constraints between the requirement groups (hereinafter referred to as “constraints”), requirement allocation, and influenced area by weighting (e.g. ASIL in ISO 26262) can be expressed.

2.4 Structure of specification

SCDL is specified in the following three chapters:

- Chapter 4: About SCDL
 - This chapter shows an outline and concept about SCDL and its roles in safety design.
- Chapter 5: Basic definition of SCDL
 - This chapter defines a basic set of required elements such as grammar and notation for SCDL.
- Appendix A: SCDL metamodel
 - This chapter shows the SCDL metamodel.

²Functional requirements e.g. intended functions, non-functional requirements, and safety requirements

³Software or hardware components

3 Relations to Other standards

3.1 Relation to Earlier Releases

The first version of the SCDL specification, Ver.1.0, was released by the Safety Concept Notation Study Group (SCN-SG). The group subsequently updated the specification up to Ver.1.5. It is assumed that SCDL will be more widely adopted and utilized in the future. Therefore, starting with this revised version 1.6.0., the SCDL specification has been transformed to ASAM and released as an ASAM international standard.

3.2 Relationship with ISO 26262

This specification supports effective and efficient implementation of the requirements related to the following specified in ISO 26262:2018.

- Part 3 Clause 7: Functional safety concept
- Part 4 Clause 6: Technical safety concept
- Part 5 Clause 6: Specification of hardware safety requirements
- Part 5 Clause 7: Hardware design
- Part 6 Clause 6: Specification of software safety requirements
- Part 6 Clause 7: Software architectural design
- Part 8 Clause 6: Specification and management of safety requirements
- Part 9 Clause 5: Requirements decomposition with respect to ASIL tailoring
- Part 9 Clause 6: Criteria for coexistence of elements
- Part 9 Clause 7: Analysis of dependent failures
- Part 9 Clause 8: Safety analyses

4 Terms and Definitions

For this document, the following terms and definitions apply. Some of the terms are defined in ISO 26262.

4.1 Assignment

Assignment of a weighting of a requirement to an element (4.2)

4.2 Element

[SEE: ISO 26262-1:2018, 3.41]

4.3 Freedom from interference

[SEE: ISO 26262-1:2018, 3.65]

4.4 Functional requirement

In the context of the specification, a functional requirement is a service expected by stakeholders. (Note:) A function describes “how something works or operates. A unique role or behavior for which each element (4.2) or part is responsible, while they are interacting each other to constitute the entirety”.

4.5 Functional safety

[SEE: ISO 26262-1:2018, 3.67]

4.6 Independence requirement

A requirement which means the absence of simultaneous violation of multiple requirements resulting from a common cause

4.7 Intended functionality

[SEE: ISO 26262-1:2018, 3.83]

4.8 Non-functional requirement

In the context of the specification, relative to a functional requirement (4.4), a non-functional requirement is a constraint for a functional requirement to be realized. For example, independence requirement (4.6) and freedom from interference (4.3) requirement. However, an independence requirement and freedom from interference requirement may become a functional requirement in the process of refinement.

4.9 Other technology

[SEE: ISO 26262-1:2018, 3.105]

4.10 Safety architecture

[SEE: ISO 26262-1:2018, 3.135]

4.11 Safety mechanism

[SEE: ISO 26262-1:2018, 3.142]

4.12 Safety requirement

Roles, functions, and behaviors relating to the safety of an element (4.2).

4.13 Weighting

Degree of criticality for safety

4.14 Abbreviations

Abbreviation	Description
ADL	Architecture Description Language
ASIL	Automotive Safety Integrity Level
DFD	Data Flow Diagram
FBD	Function Block Diagram
UML	Unified Modeling Language
SysML	Systems Modeling Language
SCDL	Safety Concept Description Language
E/E	Electric and/or Electronic

5 About SCDL

5.1 Overview

This chapter describes the background, purpose, and concept of SCDL.

5.2 Introduction and concept

Triggered by the publication of a series of ISO/IEC functional safety standards, the concept of safety design for the system, in other words, how safety architecture should be, has been a bigger topic than before. The idea is that safety architecture is very important to ensure accountability of the safety design and testability/verifiability.

In order to argue the appropriateness of the safety design, it is at least necessary that the relationship between the requirements and the system constituents is unambiguous and expressed in an easily understood, hierarchal manner.

Safety architecture is often discussed mainly based on the following three topics (1, 2, and 3).

1. Sufficient measures for possible risks

Safety measures are usually called safety mechanisms when they are implemented and effectively work in a system. SCDL addresses such safety mechanisms.

To specify safety mechanisms, the aspects described in Figure 1 should be considered.

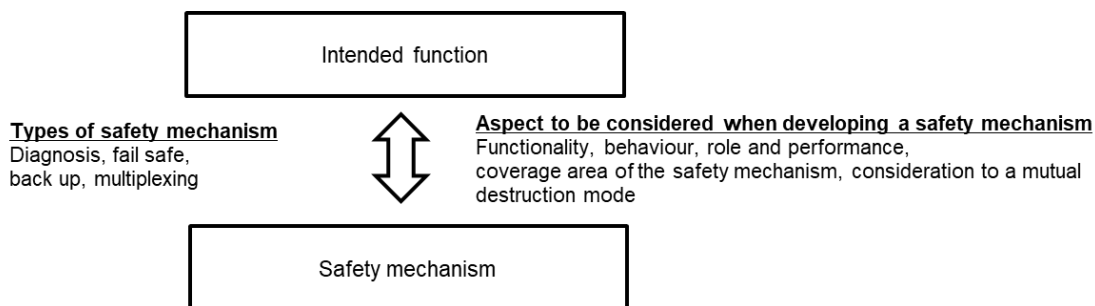


Figure 1 Specification of safety mechanism

2. Identification and separation of safety-related portions

Each safety-related portion should be handled in consideration of its importance. Arguments for safety design will become possible by identifying safety-related portions and weighting each of them.

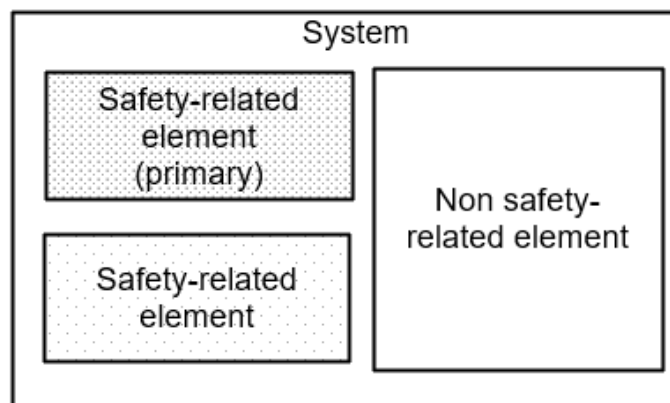


Figure 2 Identification and separation of safety-related portions

3. Systematic, structural, and hierarchical argumentation for 1 and 2

For the system to be designed, architecture is detailed out hierarchically in a top-down manner. Safety architecture is also specified in the same manner; for example, from the product concept and system concept down to hardware and software level concepts. At any level, safety design is argued by considering the structure of safety mechanisms and the identification of safety-related portions hierarchically.

Currently, there are many instances that safety architectures are specified using existing notation methods such as UML, SysML, ADL, EAST-ADL, DFD, or FBD.

However, to support the activities from consideration of safety architecture through its argumentation continuously and consistently, the existing notations and languages applied are considered to be insufficient. Unambiguous, semi-formal notation and methodology are required. It is unreasonable to expect the existing methods to meet the needs for specification of the safety architecture since they were invented and devised based on their own needs and purposes and were not prepared to deal with the safety architecture specifically and appropriately. Therefore, a safety-design-oriented semi-formal notation has been discussed and developed in an attempt to clearly illustrate the safety architecture.

In system development with limited resources, the workload spent for communication or review among involved parties should be minimized. Thus, it is important to avoid using a unique set of, or too specific, symbols for notations for safety architecture. Also, one of the important characteristics of the notation is to help engineers to intuitively understand the specifications. This way, they can focus more on communication or discussion on the safety design. This is achieved by providing this notation as an extended version of the various graphical notations with which they are already familiar.

5.3 Scope

SCDL (or a notation method, notation language) is intended to be applied for the description of specifications, i.e. to develop, verify, and reuse safety architecture effectively and efficiently. It is also assumed that SCDL is applied for architecture description of safety-related portions with some parameters related to safety design such as weighting. Depending on the technological field, safety-related requirements are sometimes categorized as non-functional requirements; but this document covers both safety-related and non-safety-related requirements as shown in Table 1.

Table 1 Scope of SCDL

	Functional requirement	Non-functional requirement
Safety-related	To be applied	May be applied as needed
Non-safety-related	May be applied as needed	N/A in principle

5.4 Basics of the SCDL notation

Based on the background previously mentioned, SCDL addresses the following to properly describe safety requirements specifications and safety architecture:

- To deal with requirements and elements (subsystems, components, etc.) separately, and at the same time express both of them in the same view.
- To illustrate interactions (such as exchange of information, signals, messages, etc.) and functional dependencies among requirements in the form of conventional DFD / FBD notations.
- To express, in a simple way, the roles of safety mechanisms with redundancy of requirements and constraints between the requirements.
- To explain the background of the derivation of safety requirements regarding safety mechanisms including safety analysis.
- To clearly show the grouping of safety requirements.
- To support the identification of targets or scope for dependent failure analysis across relevant requirement groups.
- To visualize the source of the weighting, i.e. by which requirement allocation or by which freedom from interference requirement, the weighting was determined.
- To support development consistently from the system level through the implementation levels such as the software or hardware level.
- The grammar and notation of SCDL are based on the modeling functionality realized on a software tool.

SCDL is an effective description method as a semi-formal notation required for safety architecture in ISO 26262:2018. Also, the specification information of the safety architecture developed with SCDL realizes efficient safety architecture description and safety analyses with the support of software tools.

6 Basic definition of SCDL

6.1 Overview

In this chapter, definitions of symbols, the meaning of combinations of the symbols, and details of categories of structure diagrams are described as the definition of SCDL.

- 6.2 “Definition of symbols” explains requirement, requirement group, redundancy of the requirement groups, element, interaction, and system boundary interaction.
- 6.3 “Meaning of combinations of symbols” explains: allocation of a requirement to an element, the way to group requirements, how to derive constraints at pairing, notation for freedom from interference requirement, branching of interaction lines between requirements

6.2 Definition of symbols

This section defines basic symbols (requirement, element, interaction of them, and freedom from interference) used for the fundamental notation of SCDL. Since line type, thickness, and colour (such as fill colour, line colour, and font colour) of all symbols are optional, users of SCDL should specify rules to distinguish different symbols.

SCDL defines the following constituents:

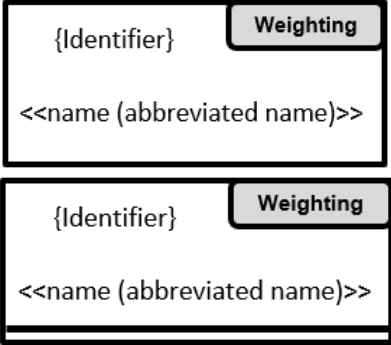
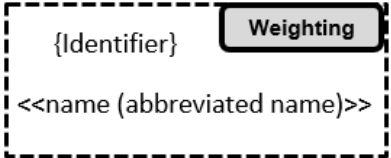
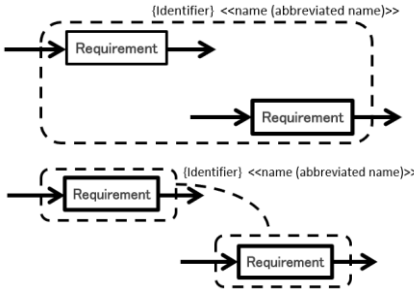
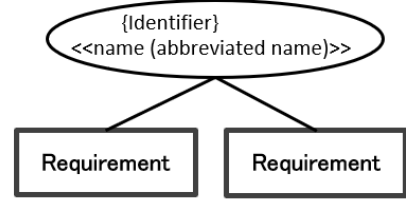
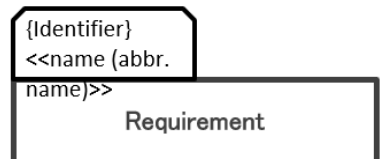
- Requirement
- Requirement group
- Element

SCDL also defines the relationship among the following

- Interaction
- System boundary interaction
- Freedom from interference arrow
- Requirement group pairing line
- Connecting line from constraint





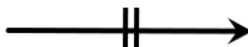


Notation of the constituents is shown in Table 2. Symbols such as “{ }”, “<< >>”, or “()” in the table indicate the entry fields for each item; and they do not need to be described in figures.

Table 2 SCDL basic definition (constituents)

Name of symbol	Symbol	Description
Requirement		<p>A requirement is expressed in a rectangle and its weighting is shown in a small rectangle located at the upper right corner of the larger rectangle.</p> <p>An abbreviated name is to help users to understand the name of the constituent.</p> <p>If the destination of requirement allocation in an element has not been decided or if the requirement cannot be allocated, a double line is used for the base of the rectangle as shown in the lower figure.</p>
Element		<p>An element is expressed in a rectangle and its weighting is shown in a small rectangle located at the upper right corner of the larger rectangle.</p> <p>Different line types or colours should be used to distinguish an element and a requirement.</p>
Requirement group	<p>1.</p>  <p>2.</p>  <p>3.</p> 	<p>There are three options for notation to describe requirement groups. Any option may be chosen.</p> <ol style="list-style-type: none"> 1. Enclosure type: enclose multiple requirements which belong to the same group in a frame line. Multiple frames may be connected with a dotted line if they belong to the same group. 2. Balloon type: draw an oval shape to express a requirement group. Connect it to requirements which belong to the same group with straight lines. 3. Tab type: add a tab on the upper side of a rectangle and describe the requirement group to which the requirement belongs. In the tab, describe an identifier to indicate the same requirement group to which requirements belong.

Relationship notations are shown in Table 3.

Table 3 SCDL basic definition (relationship)

Name of symbol	Symbol	Description
Interaction	 {Identifier} <<name (abbreviated name)>>	Use a one-direction arrow between two requirements to show "Interaction".
System boundary interaction	 {Identifier} <<name (abbreviated name)>>	Use a one-direction block arrow between the external and a requirement to show "System boundary interaction".
External plant	 {Identifier} <<name (abbreviated name)>>	Use a five-point star to show "External plant".
Other technology links	 {Identifier} <<name (abbreviated name)>>	Place a small filled-in circle on a one-direction block arrow (System boundary interaction) to show "Other technology link". Connect small filled-in circles which are linked to each other with a line segment.
Freedom from interference	 {Identifier} <<name (abbreviated name)>>	Use a line arrow and two intersecting parallel lines to show "Freedom from interference". Place parallel lines between the start point and the endpoint of the line arrow.
Requirement group pairing	 {Identifier} <<name (abbreviated name)>>	Use a dashed line arrow (double) to connect requirement groups to show "Requirement group pairing".
Connecting line	 {Identifier} <<name (abbreviated name)>>	Use a "Connecting line" with diamond-shaped endpoints.

6.2.1 Notation of requirements

A requirement is expressed in a rectangle, where functionality, role, and behaviour are provided. One rectangle represents one requirement. The naming convention for the description is not defined in SCDL. The line type of the rectangle for a requirement is arbitrary, however, the rectangle for a requirement and the rectangle for an element which is defined in the next section shall be distinguished.

The rectangle to express a requirement has an entry field for weighting. The field for weighting shall be a smaller rectangle, normally placed on the upper-right corner of the requirement, whose two sides are inscribed in the larger rectangle of the requirement. The weighting shall be described according to the rules specified in relevant safety standards. However, it is not to be described until the weighting is assigned to the requirement.

Normative items

- Basic
 - Describe either “ID” or “name (abbreviated name)”, or both as an identifier.
 - Description of weighting may be omitted.
- Position and shape
 - The shape shall be a rectangle.
 - The line type for the rectangle which expresses a requirement shall be a solid line.
 - The field for weighting shall be a smaller rectangle, normally placed on the upper-right corner of the requirement, whose two sides are inscribed in the larger rectangle of the requirement.
 - If an element is placed in combination with a requirement, they shall not be inscribed / circumscribed / intersected with each other.

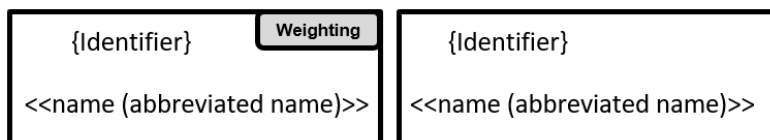
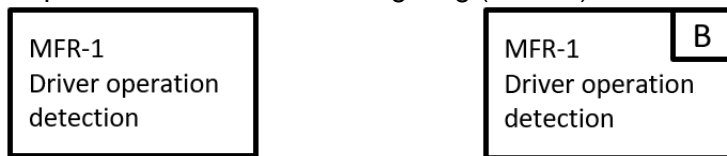


Figure 3 Notation of requirements

Example:

Left: a requirement for which the weighting (ASIL) has not been determined yet

Right: a requirement for which the weighting (ASIL B) has been defined



6.2.2 Notation of interactions

The exchange of information / signals / messages between requirements in SCDL is called interaction. Representative notation for interactions is shown in Figure 6.

A rectangle which expresses a requirement has an interaction represented by one or more inputs and one output. This is because requirements allocated to the architecture have dependencies between them and such dependencies are expressed as interactions.

An interaction is expressed by an “arrow”. The “endpoint without an arrow” is connected to the origin of a requirement and the “endpoint with an arrow” is connected to the target of the requirement.

As shown in Figure 5, interaction starting from the “endpoint without an arrow” may be branched halfway and connected to multiple requirements. This is because what a requirement specifies is referred to and interacts with other multiple requirements.

However, inputs to a requirement shall not be joined by connecting interactions halfway. See Figure 4. The reason is that a certain intent should exist for inputs merge and the intent should be clarified as a requirement. Therefore, inputs of multiple interactions should be described for the clarified requirement.

Normative items

- Basic
 - Describe either “ID” or “name (abbreviated name)”, or both as an identifier.
 - A requirement shall have one or more (multiple) interactions (inputs). (See Figure 4)
 - If the same information (interaction) is shared by two or more requirements at a lower level, describe them in such a way that the interaction is branched. (See Figure 5)
- Position and shape
 - The shape shall be a line arrow.
 - Both endpoints of the line shall touch requirements. Contact points for the interactions and the requirements shall be on the sides of the requirements as a start point and an endpoint.
 - The line arrow of the interactions may be intersected.



Figure 4 Example of the case with multiple input interactions



Figure 5 Example of branching

Example

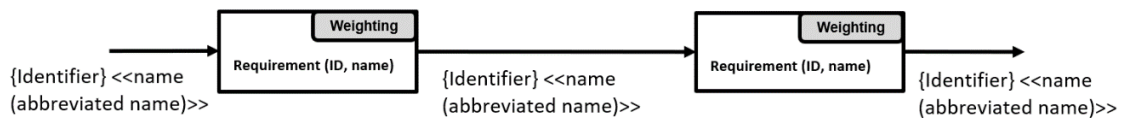


Figure 6 Representative notation of requirements and interactions between requirements

6.2.3 Notation for system boundary interactions

Interactions which represent an exchange of mechanical dynamics between requirements (hydraulic, water volume, etc.) are called system boundary interactions. Notation of system boundary interactions is shown in Figure 7.

A system boundary interaction is expressed with a “block arrow”. To express an input, the “endpoint with an arrow” is connected to a requirement, and the “endpoint without an arrow” is connected to another requirement outside the system boundary. To express an output, the “endpoint without an arrow” is connected to a requirement, and the “endpoint with an arrow” is connected to another requirement outside the system boundary. However, a description of the connection with the requirement outside the system boundary may be omitted. Either one of the endpoints does not contact the requirement in this case. Even when the description of the connection is omitted, the endpoint which is not connected to the requirement shall still be placed outside the system.

Normative items

- Basic
 - Describe either “ID” or “name (abbreviated name)”, or both as an identifier.
 - One end of the line shall be connected to a requirement.
- Position and shape
 - The shape shall be a “block arrow”.
 - The block arrow shall be thick enough to be distinguished from an interaction.

Example

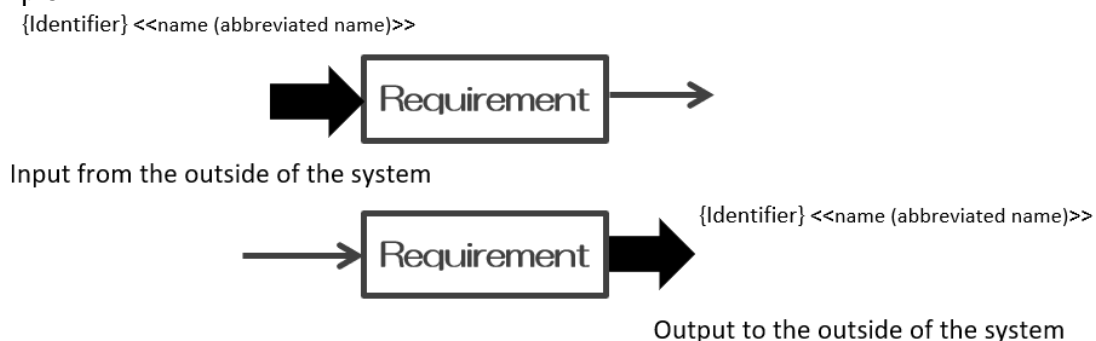


Figure 7 Example of notation for system boundary interactions

6.2.3.1 External plant⁴

When control results are monitored with a sensor, or when the control results are reflected the behaviour of the system and measured by sensing, use a 5-point star mark to express connection or interaction outside the system and the item controlled, which are subject to notation.

The external plant is a derivative notation of the system boundary interaction. Also, the external plant shall be used only for the system boundary interaction, because it is a notation to express connection or interaction outside the system which is subject to notation and to express a relationship between inputs and outputs to the controlled item.

Normative items

- Basic
 - Describe either “ID” or “name (abbreviated name)”, or both as an identifier.
- Position and shape
 - External plants shall be described on the system boundary interaction.
 - The shape shall be a 5-point star.

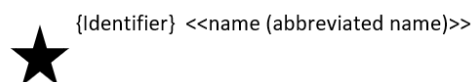


Figure 8 External plant

Example

The following figure shows that control results are reflected to behaviour of the control target in Element 1-4, and the behaviour is measured by sensing in Element 1-2.

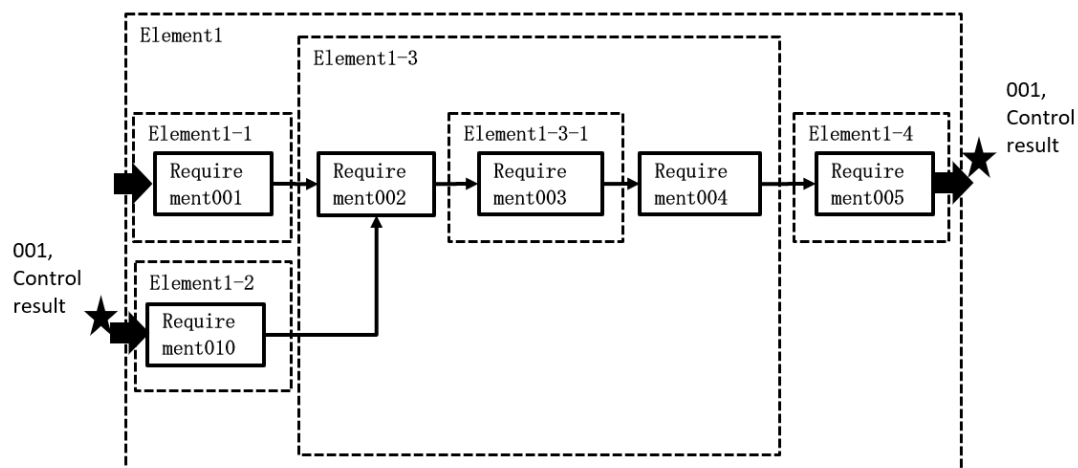


Figure 9 Example of use of external plant

⁴ “External” means outside the E/E systems.

6.2.3.2 Other technology links⁵

The notation for inputs linked by mechanical architecture etc. is shown below.

Other technology links are a derivative notation of the system boundary interaction.

Normative items

- Basic
 - If an “input” or “output” system boundary interaction is linked mechanically, the connection with the destination of the link shall be described.
 - If the inputs and/or outputs have multiple destinations for the link, the connection shall be made to all.
 - Describe either “ID” or “name (abbreviated name)”, or both, as an identifier.
- Position and shape
 - To clearly determine mechanical links outside the system, the notation shall be as follows.
 - If the input or output side is linked, place filled-in circles so that the circles touch the “endpoints with arrows” of all the linked system boundary interactions and connect the circles using a line segment.
 - The circles shall be clearly recognized as circles for both input and output sides.

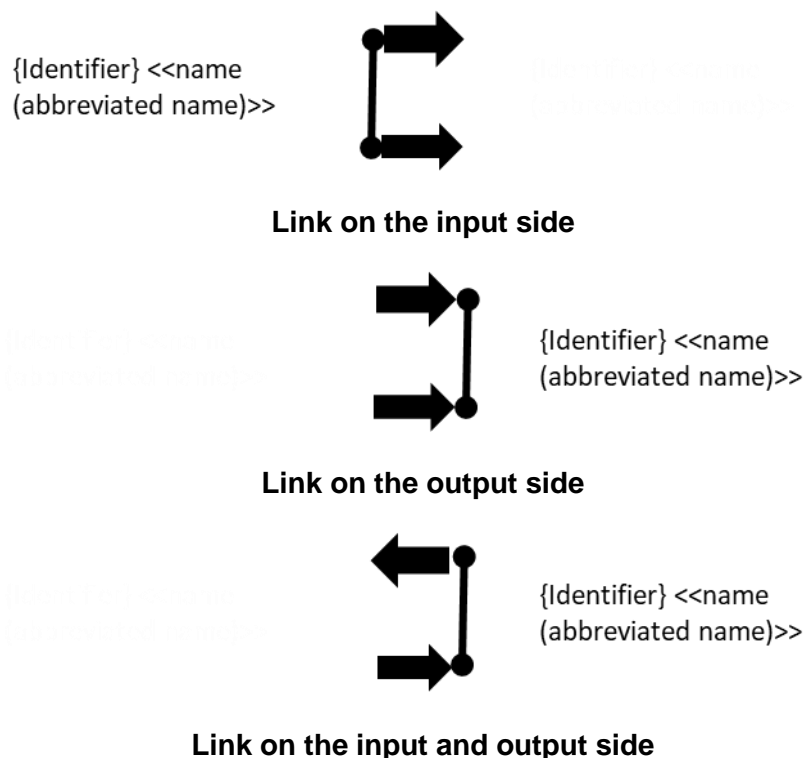


Figure 10 Notation for Other technology links

Example

Figure 11 shows an example in which a sensor is linked mechanically.

⁵ "Other technology" means technologies other than those of the E/E systems.

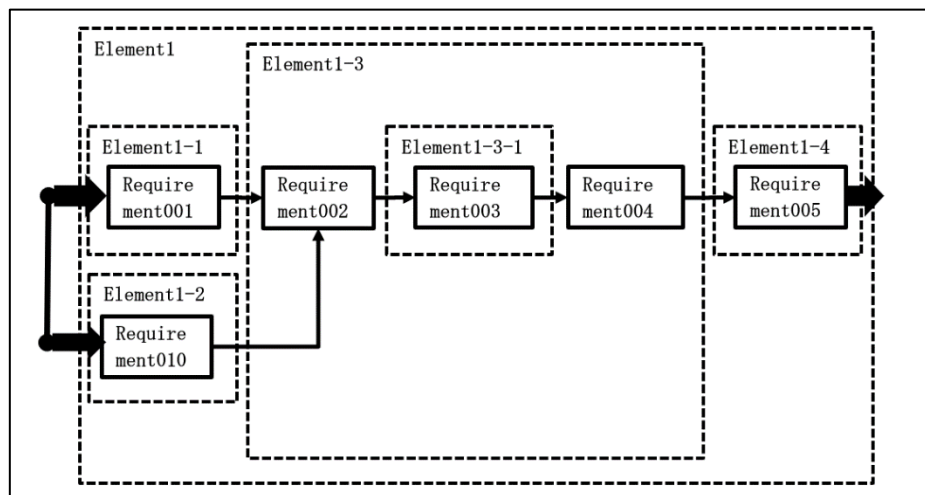


Figure 11 Example of use of Other technology links

6.2.4 Prohibitions and exceptions for interaction notation

Typical prohibitions and exceptions are described for notation of interaction lines between multiple requirements.

Prohibitions

- 1 Two or more output interactions from one requirement shall be avoided because the requirement would become a non-atomic requirement. (See Figure 12)
- 2 Since the logic of merged interactions itself should be specified as a separate requirement, multiple-input interactions shall not be expressed as merged. (See Figure 13)



Figure 12 Prohibition: Avoid two or more output interactions from one requirement



Figure 13 Prohibition: Input interactions shall not be merged

Exception

1. For items/systems/subsystems, interaction generally has a route connected from input through output. However, a requirement without inputs or outputs exists as an exception (e.g. initial value setting, random numbers generation, reset).



Figure 14 Example of a requirement without inputs



Figure 15 Example of a requirement without outputs

6.2.5 Notation for elements

Elements represent systems, subsystems, components, units, modules, parts, and circuit blocks, etc., as well as their nested structures.

Use a rectangle to describe an element. The line type shall be dashed lines (straight lines made up of dots or dashes at regular intervals) so that the elements can be distinguished from rectangles for the requirements.

The internal structure of the higher-level element can easily be expressed by placing lower-level elements as both subsets within the higher-level element and as subsets within other lower-level elements.

Typical notation of elements in SCDL is shown in Figure 16.

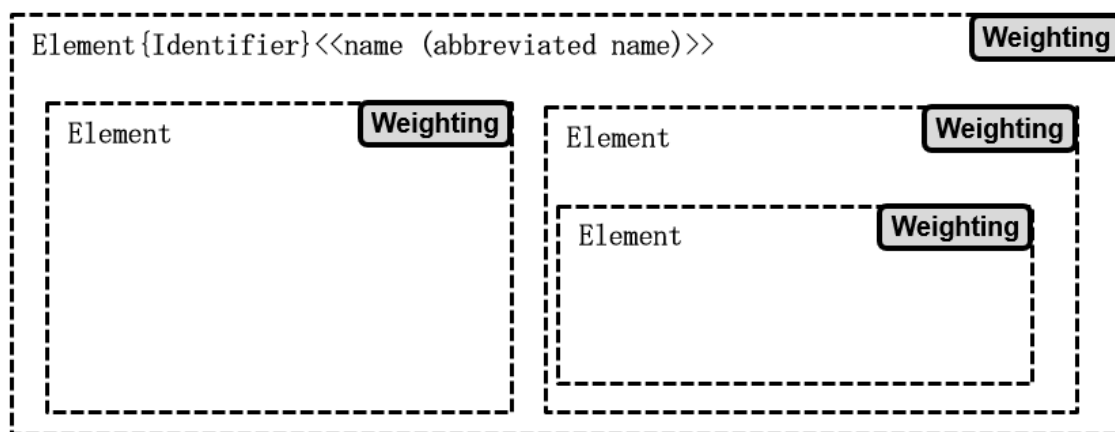


Figure 16 Notation for elements

Normative items

- Basic
 - Elements represent systems, subsystems, components, units, modules, parts, and circuit blocks, etc., as well as their nested structures.
 - For the elements, describe either “ID” or “name (abbreviated name)”, or both as an identifier.
- Position and shape
 - Use a rectangle to describe an element.
 - Element has a rectangular entry field to describe the weighting.
 - The line type of the element shall be dashed lines (straight lines made up of dots or dashes at regular intervals) to distinguish from the line type for a requirement.
 - The field for weighting shall be a smaller rectangle, placed on the upper-right corner of the element. The two sides (the top side and the right side) of the field are inscribed in the larger rectangle of the element.
 - The size of the rectangle shall be large enough to make the described weighting visible.
 - To notate a nested structure, sub-elements shall be described inside the parent element using the same rectangle shape, but with a smaller size of rectangle than the parent element. And they shall be placed without contacting each other. The outermost element represents the system boundary.
 - An element shall not be divided into two elements by partitioning it by a dashed line. (See Figure 17)
 - Elements shall be described as a complete diagram. They shall not partially overlap. (See Figure 18)
 - A higher-level element and a lower-level element shall not be inscribed. (See Figure 19)

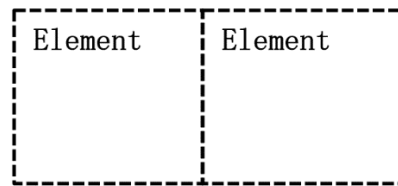


Figure 17 **Example of prohibition when describing elements (division)**

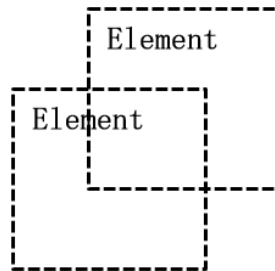


Figure 18 **Example of prohibition when describing elements (overlap)**

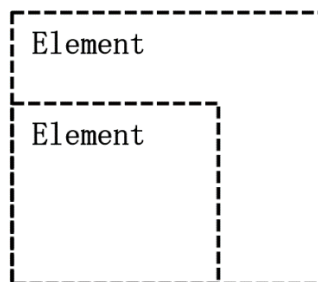


Figure 19 **Example of prohibition when describing elements (inscribing)**

6.2.6 Constraints

Constraints which are the restrictions for realizing functional requirements can be expressed as requirements. Requirements for independence or freedom from interference, which become constraints for requirements grouping or between groups, are addressed. However, such requirements for independence or freedom from interference may become functional requirements in the process of refinement.

The following normative items are specific to constraints.

Normative items

- Basic
 - Constraints are described as requirements.
 - The relationship between requirement groups is described by a constraint.
 - If the destination of constraints allocation in an element has not been decided or cannot be allocated, double the base of the rectangle as shown in Figure 20.



Figure 20 **Notation of constraints when the destination in an element has not been decided**

- Position and shape
 - Constraints shall be described as requirements which are connected by pairing lines or connecting lines from freedom from interference.

6.2.7 Connecting lines

A connecting line is used to connect a pairing line/ freedom from interference arrow and a constraint.

Normative items

- Basic
 - Describe either “ID” or “name (abbreviated name)”, or both as an identifier.
 - One end of the line segment shall be connected to a requirement. The other end shall be connected to another line (notation of freedom from interference or a pairing line)
- Position and shape
 - To represent a connecting line, a line segment whose both endpoints are diamond shall be used.

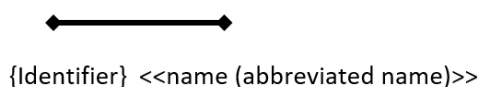


Figure 21 Notation of connecting line

Example

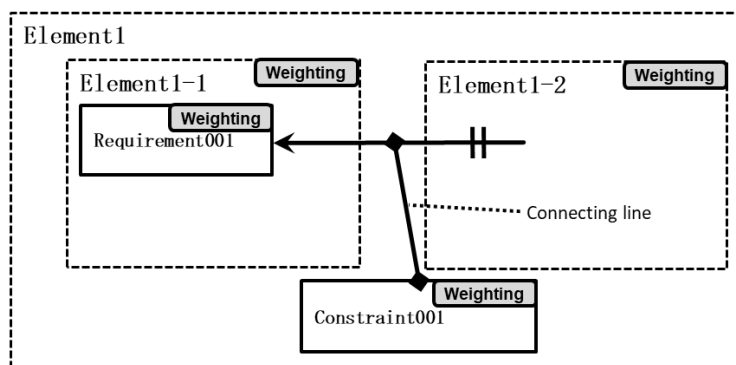


Figure 22 Example of a connecting line connected with freedom from interference

6.3 Meaning of combinations of symbols

SCDL uses combinations of the elements defined. Typical combinations of notation are shown in this section.

6.3.1 Allocation of requirements to elements

Requirements allocation to elements is specified below.

Normative items

- Basic
 - Requirements shall be allocated (described) to the element which realizes the requirements (See Figure 23).
 - Multiple requirements may be allocated (described) to one element.
 - If requirements with weighting assigned are allocated to an element, weighting for the element shall be assigned accordingly based on the rules defined in various safety standards. However, the weighting for the element is not described until the weighting for the associated requirements is assigned (See Figure 24).

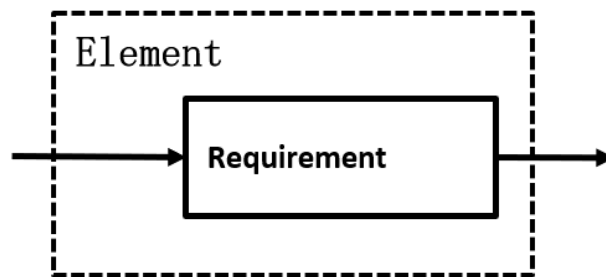


Figure 23 Description of allocation of requirement and element

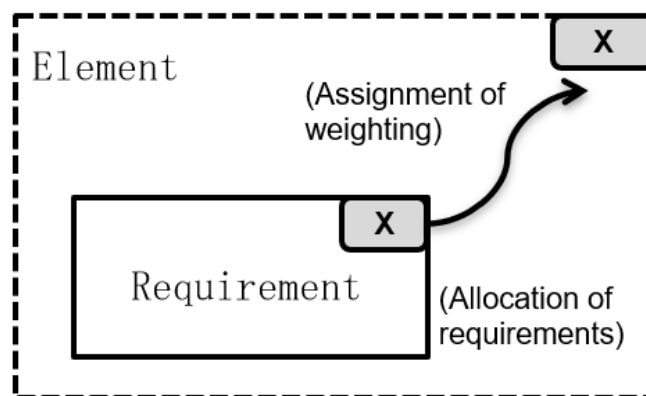


Figure 24 Assignment of weighting for element

- Position and shape
 - To clearly distinguish between the elements to which requirements are allocated and interactions between the requirements, a notation shall be as follows:
 - Requirement allocated to an element shall be described inside the element without contacting the borderline of the element. (See Figure 23)
 - If multiple requirements are allocated to an element, the requirements shall not contact each other.
 - Requirements shall not be inscribed or intersect with the borderline of an element. (See Figure 25 and Figure 26)
 - Even when an element is nested, requirements shall not be inscribed, circumscribed, or intersect with any of the borderlines of a higher-level element and a lower-level element. (See Figure 27)

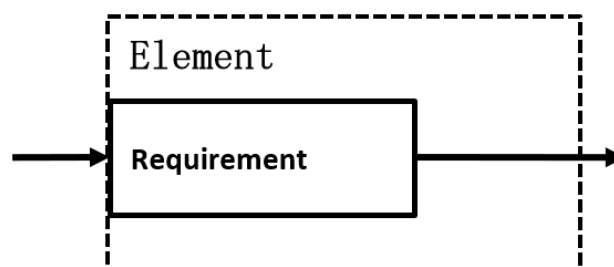


Figure 25 Example of prohibition of inscribing

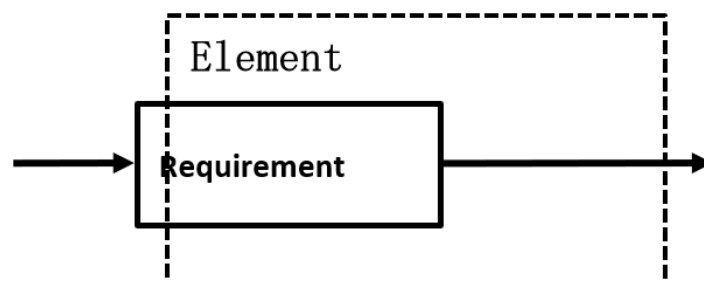


Figure 26 Example of prohibition of intersecting

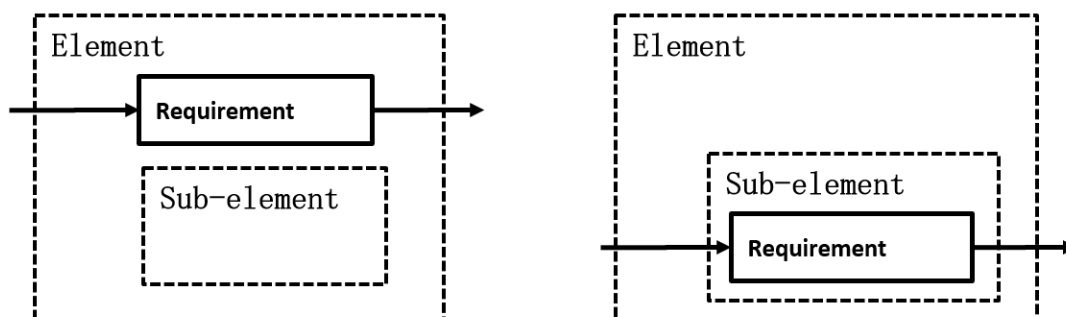


Figure 27 Example of adequate requirement allocation to the nested element

6.3.2 How to make requirement groups

When describing one or more requirement(s) collectively as one requirement, group the requirement(s). One of the three types of notation; “Enclosure type”, “Balloon type”, or “Tab type”, is used to describe the requirement group. (See Table 2) When describing a combination of redundant requirement groups, use requirement group pairing.

6.3.2.1 Requirement groups

6.3.2.1.1 Enclosure type

The “Enclosure type” has two approaches for notation: “Framed requirement group” which requirements belonging to the same requirement group are enclosed in a frame line, and, “Connected requirement group” which requirements belonging to the same requirement group are connected by a line.

a) Framed requirement group

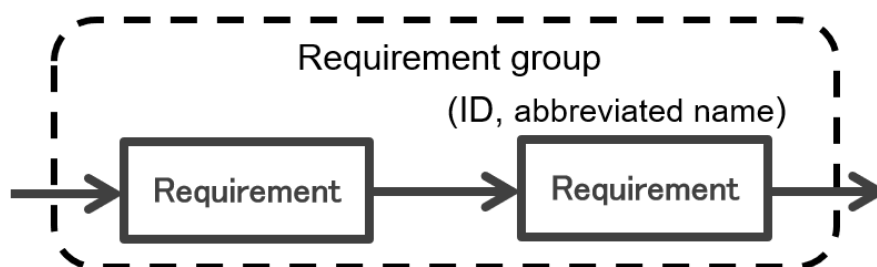


Figure 28 Notation for framed requirement group

Normative items

- Basic
 - Multiple requirements are grouped in a frame line and identified as a “requirement group”.
 - To identify a requirement group, describe either “ID” or “name (abbreviated name)”, or both as an identifier.
- Position and shape
 - Use a rectangle or a polygon for notation. The frame line should be easy to identify requirement groups.
 - The frame shall be distinguishable from the notation of elements. Since this notation for a requirement group is similar to that for an element, a rectangle with rounded corners or a dashed line should be used to distinguish between them. The thickness or colour of the line may be changed, also the inside of the frame may be filled with colour for differentiation.

b) Connected requirement group

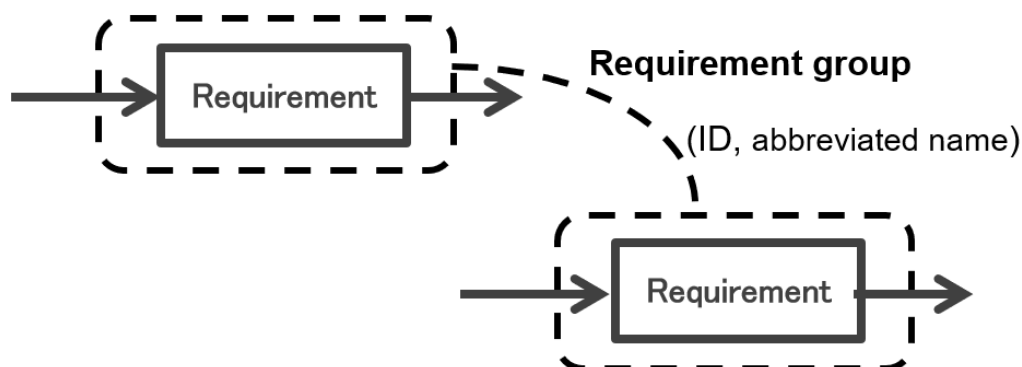


Figure 29 Notation for connected requirement group

Normative items

- Basic
 - The same requirement groups are connected by a bridging line. When requirement groups are distantly positioned and cannot be grouped by a frame line, a bridging line shall be used to identify them as the same requirement group.
 - For framed requirement groups, follow the normative items of the “Framed requirement group”.
 - To identify a requirement group, describe either “ID” or “name (abbreviated name)”, or both as an identifier.
- Position and shape
 - Both endpoints of a bridging line shall be circumscribed to frame lines of requirement groups. However, the line shall not protrude through the frame lines of the requirement groups.

6.3.2.1.2 Balloon type

Describe requirement groups using an oval. Connect a requirement group and the requirement(s) belonging to the requirement group by bridging lines.

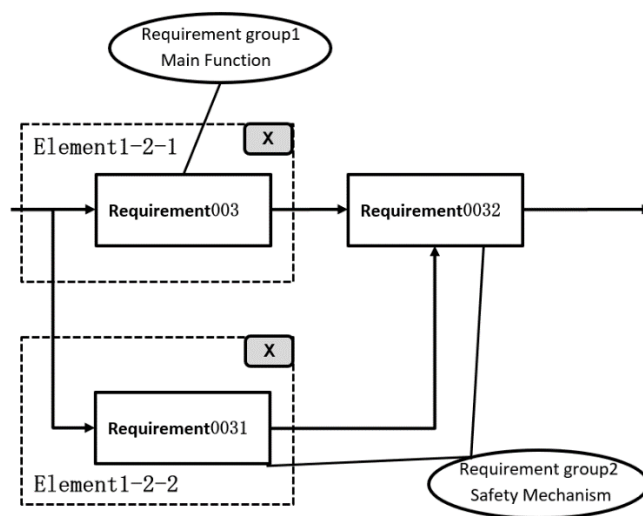


Figure 30 Notation for “Balloon-type” requirement group

Normative items

- Basic
 - Requirements which belong to the same requirement group are connected by a bridging line. When requirements which are distantly positioned cannot be grouped by a frame line, a bridging line shall be used to identify them as the same requirement group.
 - To identify a requirement group, describe either “ID” or “name (abbreviated name)”, or both as an identifier.
- Position and shape
 - Use an oval to describe a “Balloon-type” requirement group.
 - One endpoint of a bridging line shall be circumscribed to a rectangle of a requirement, and the other shall be circumscribed to a frame line of a requirement group. However, the bridging line shall not protrude through the rectangle of the requirement or the frame line of the requirement group.

6.3.2.1.3 Tab type

Describe a tab on the upper side of a rectangle which expresses a requirement and notate a requirement group to which the requirement belongs.

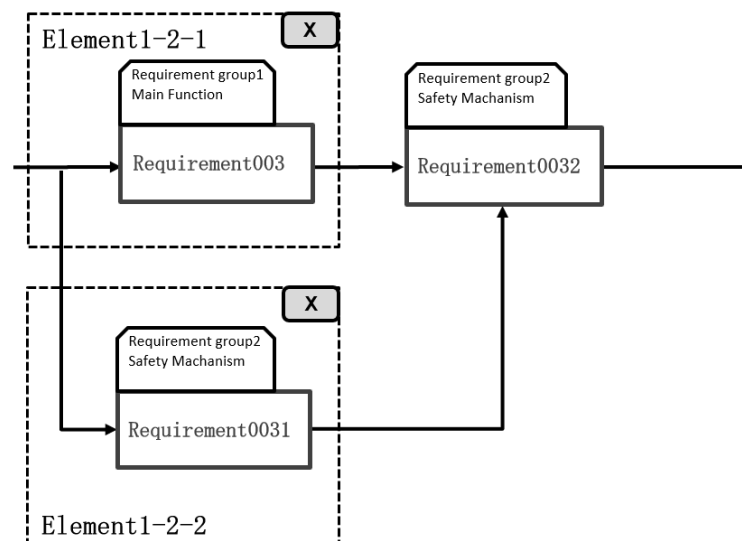


Figure 31 Notation for “Tab-type” requirement group

Normative items

- Basic
 - Add the same tab to the requirements to be grouped as a requirement group. When requirements which are distantly positioned cannot be grouped by a frame line or a bridging line, the same tab shall be used to identify them as the same requirement group.
 - To determine the requirement group to which requirements belong, describe either “ID” or “name (abbreviated name)”, or both as an identifier.
- Position and shape
 - Use a tab which is described on the upper side of a rectangle of a requirement.
 - The tab shall be circumscribed to a rectangle of a requirement. However, the tab shall not protrude through the rectangle of the requirement.
 - The tab shall not be described so that it is circumscribed to or overlaps with descriptions of other requirements or requirement groups.

6.3.2.2 Requirement group pairing

A combination of two requirement groups is called requirement group pairing. Requirement group pairing is shown by connecting two requirement groups with a dashed line arrow. Notation of requirement group pairing is explained for each notation type of the requirement group.

6.3.2.2.1 Enclosure type

For a combination of requirement groups of “Enclosure type”, notation of requirement group pairing shall be used based on each type of “Enclosure-type” notation (see 6.3.2.1).

a) Combination of framed requirement groups

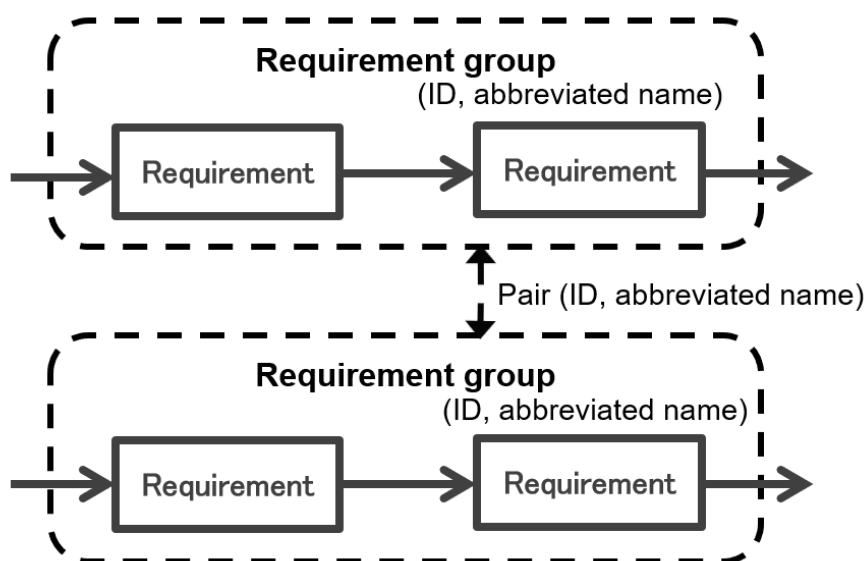


Figure 32 Combination of framed requirement groups

Normative items

- Basic
 - Two requirement groups shall be connected by a requirement pairing line. One requirement pairing line represents one combination.
 - Describe either “ID” or “name (abbreviated name)”, or both as an identifier.
- Position and shape
 - Use a bidirectional dashed line arrow.
 - Both endpoints of the bidirectional line arrow shall be circumscribed to frame lines of requirement groups. However, the line arrow shall not protrude through the frame lines of the requirement groups.

b) Combination of connected requirement groups

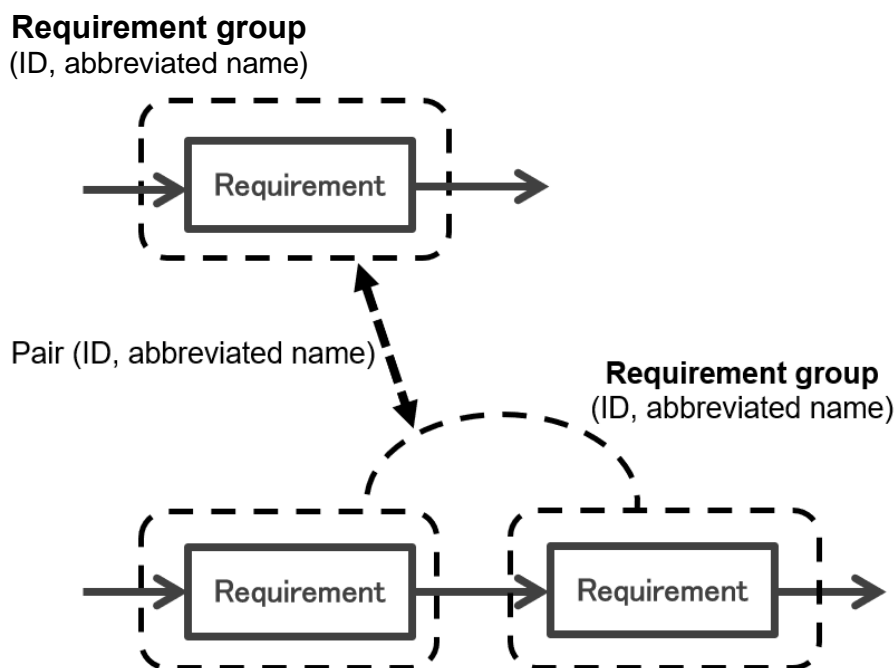


Figure 33 Combination of connected requirement groups

Normative items

- Basic
 - Two requirement groups shall be connected by a requirement pairing line. One requirement pairing line represents one combination.
 - Describe either “ID” or “name (abbreviated name)”, or both as an identifier.
- Position and shape
 - If the connection destination of an endpoint with an arrow is a frame line of a requirement group, the endpoint of the line arrow is circumscribed to the frame line of the requirement group. However, the line arrow shall not protrude through the frame line of the requirement group.
 - If the connection destination of an endpoint with an arrow is a bridging line of a requirement group, the endpoint of the line arrow is circumscribed to or overlaps with the bridging line of the requirement group.

6.3.2.2.2 Balloon type

Notation of combination of “Balloon-type” requirement groups is shown below:

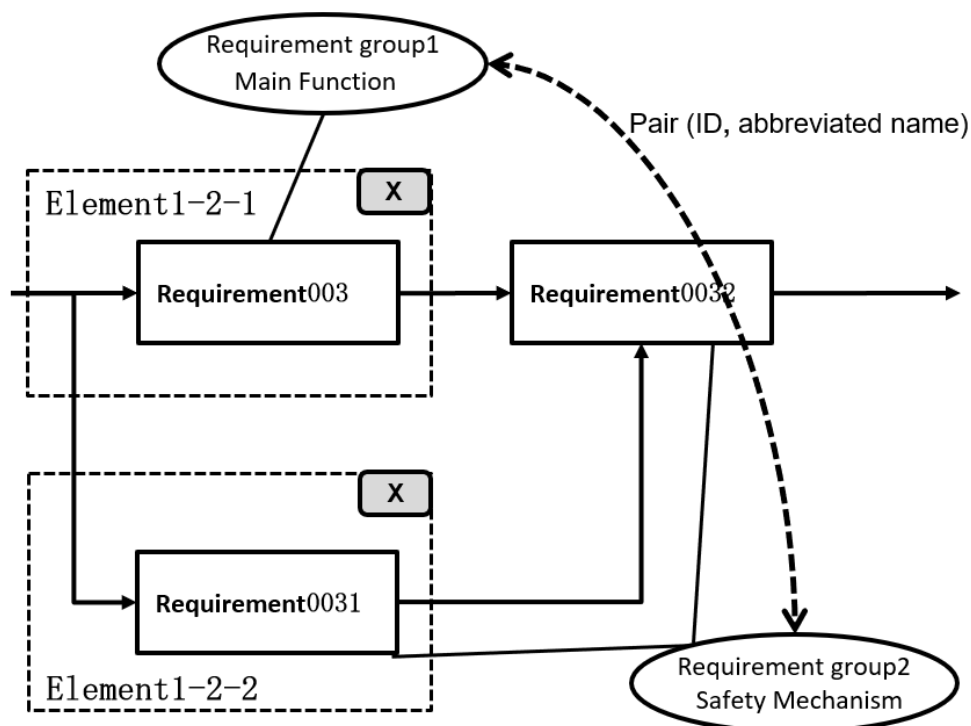


Figure 34 Notation for “Balloon-type” requirement group

Normative items

- Basic
 - Two requirement groups shall be connected by a requirement pairing line. One requirement pairing line represents one combination.
 - Describe either “ID” or “name (abbreviated name)”, or both as an identifier.
- Position and shape
 - Use a bidirectional dashed line arrow.
 - Both endpoints of the bidirectional line arrow shall be circumscribed to ovals of requirement groups. However, the line arrow shall not protrude through the ovals of the requirement groups.

6.3.2.2.3 Tab type

Notation of combination of “Tab-type” requirement groups is shown below:

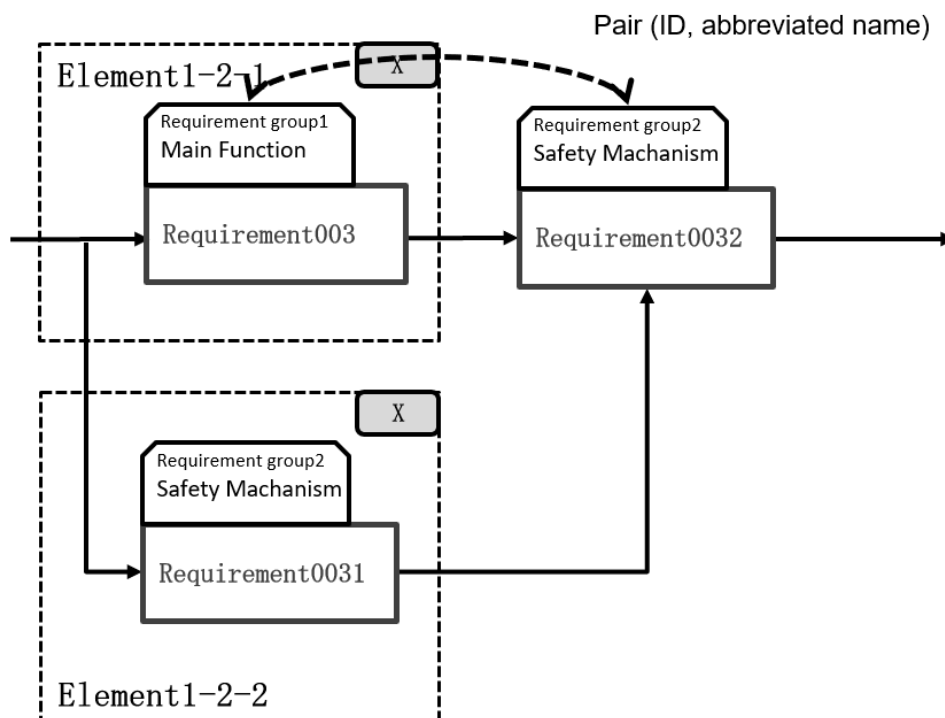


Figure 35 Notation for “Tab-type” requirement group pairing

Normative items

- Basic
 - Two requirement groups shall be connected by a requirement pairing line. One requirement pairing line represents one combination.
 - Describe either “ID” or “name (abbreviated name)”, or both as an identifier.
- Position and shape
 - Use a bidirectional dashed line arrow.
 - Both endpoints of the bidirectional line arrow shall be circumscribed to tabs of requirement groups (See 6.3.2.1). However, the line arrow shall not protrude through the tabs of the requirement groups.

6.3.3 How to describe constraints by using connecting lines from the pairing line between requirement groups

Notation described in this section is intended to address decomposition specified in ISO 26262.

6.3.3.1 Description of constraints for pairing between redundant requirement groups

For describing redundant requirement groups by “Enclosure type”

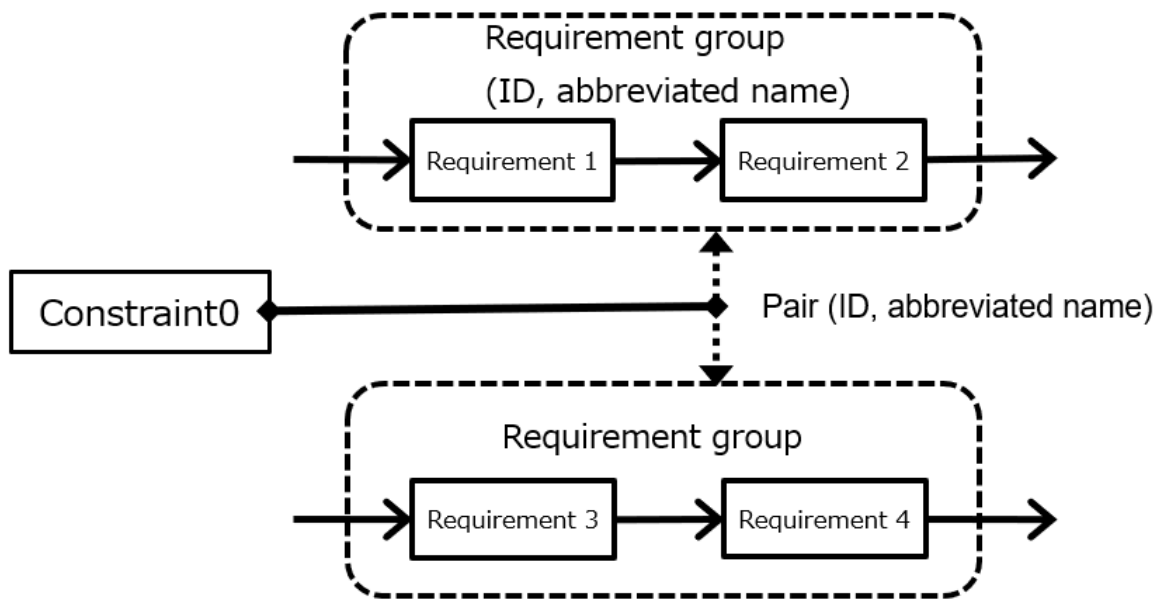


Figure 36 Notation for constraints for pairing

Normative items

- Basic
 - Constraints are described as requirements.
 - Relation between requirement groups is discussed considering only one constraint.
 - Constraints may be omitted.
 - If the destination in an element has not been decided when allocating constraints, or if constraints cannot be allocated, double the base of the rectangle as shown below.



Figure 37 Constraints

- Position and shape
 - Constraints shall be described using connecting lines from pairing lines.

For notation of requirement groups and constraints, the following notation may also be used.

6.3.3.2 Description of constraints between “Balloon-type” requirement groups

For describing redundant requirement groups by “Balloon type”

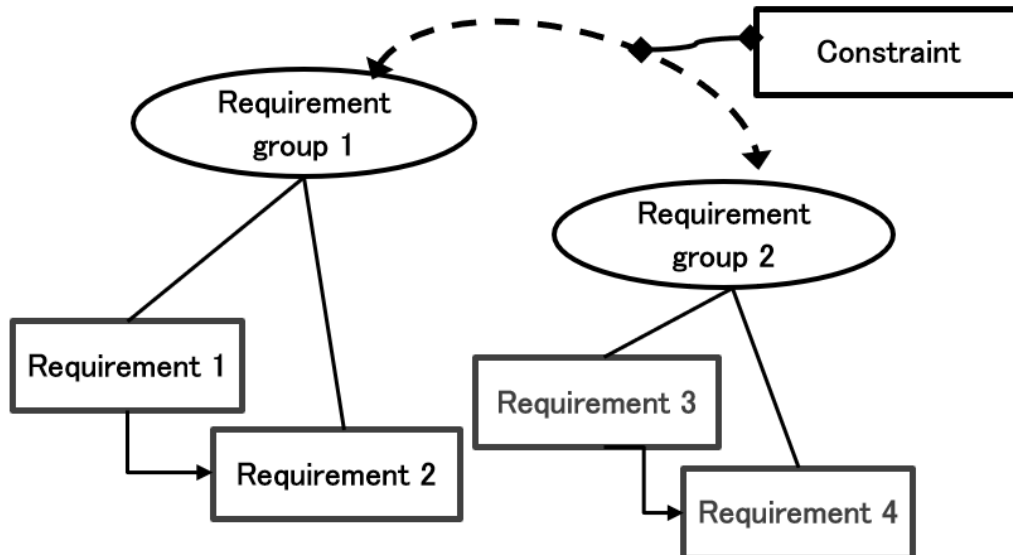


Figure 38 Notation for constraints by “Balloon type”

Normative items

- Basic
 - Constraints are described using the notation of requirements and connecting lines from bidirectional line arrows which represent requirement group pairing.
- Position and shape
 - Use connecting lines with “diamond-shaped endpoints”.
 - One endpoint of the connecting line is circumscribed to or overlaps with the line of requirement group pairing.
 - The other endpoint is circumscribed to or overlaps with the rectangle of constraints.

6.3.3.3 Description of constraints between “Tab-type” requirement groups

For describing redundant requirement groups by “Tab type”

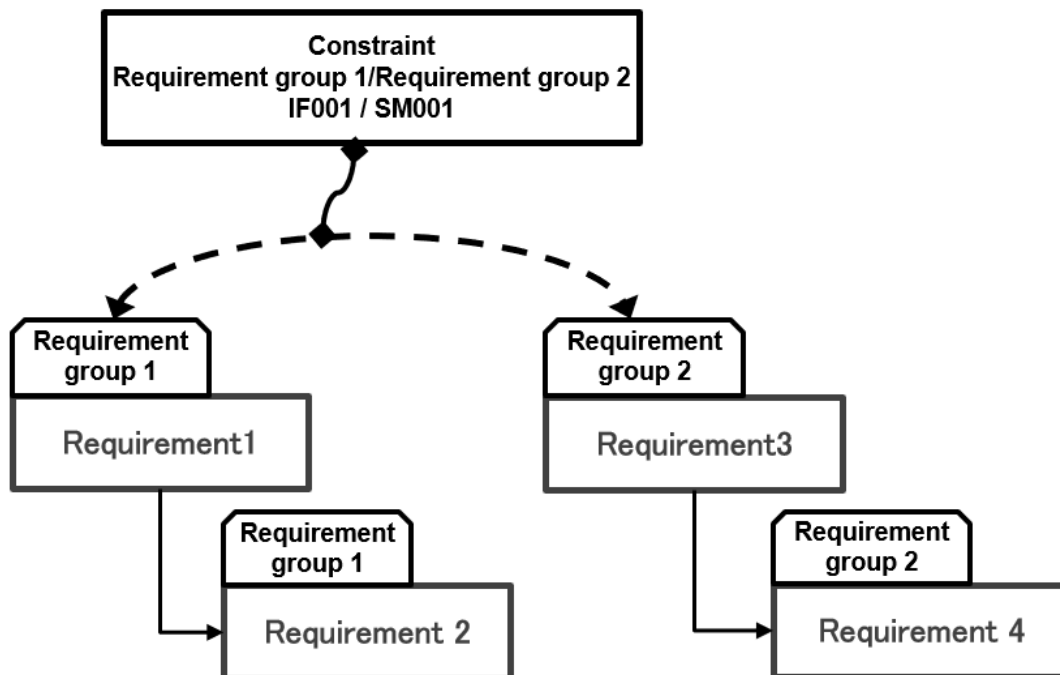


Figure 39 Notation for constraints by “Tab type”

Normative items

- Basic
 - Constraints are described using the notation of requirements and connecting lines from bidirectional line arrows which represent requirement group pairing.
- Position and shape
 - Use connecting lines with “diamond-shaped endpoints”.
 - One endpoint of the connecting line is circumscribed to or overlaps with the line of requirement group pairing.
 - The other endpoint is circumscribed to or overlaps with the rectangle of constraints.

6.3.4 Detailing constraints between requirement groups

Constraints between requirement groups can be detailed into constraints between individual requirements which belong to the two requirement groups. When detailing them, combinations of constraints among all requirements shall be considered in principle. Figure 40 shows the case in which “Constraint 0” in Figure 36 is detailed.

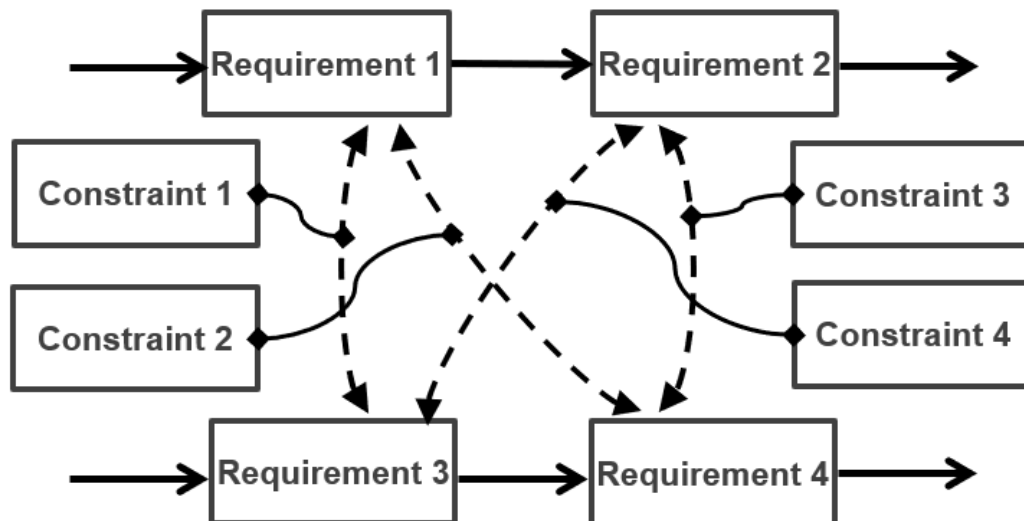


Figure 40 Detailing constraints

6.3.5 Notation for freedom from interference

Typical notation of “freedom from interference” is shown in Figure 41, Figure 42, and Figure 43. “Freedom from interference” can be described between constituents, when there is a requirement for that from an element to a requirement (Figure 41), to a requirement group (Figure 42), as well as a requirement for that between all requirements and elements which are included in another element (Figure 43). Notation of “freedom from interference” represents that there is a non-functional requirement that the element at the starting point of the arrow shall not violate the requirement at the ending point of the arrow.

If describing “freedom from interference” from an element to all requirements included in other elements, arrows of “freedom from interference” should be described from the element at the starting point to all the requirements included in another element at the ending point of the arrow of “freedom from interference”. However, in a simplified manner, an arrow of “freedom from interference” may be described between elements at the starting point and at the ending point.

Requirements for “freedom from interference” are described as the requirements led by a connecting line from the “freedom from interference” notation (as shown below; Constraint 001). In this case, the requirements of constraints are allocated in the higher-level element to which the requirements for “freedom from interference” belong.

Normative items

- Basic
 - A line arrow and two parallel lines intersecting with the line arrow shall be used as a symbol.

- Constraints are described as requirements.
- Position and shape
 - Two parallel lines are described intersecting with a line arrow between the starting point of the arrow and the ending point of the arrow.
 - The endpoint of the arrow (end) shall be an arrow.
 - The starting point of the arrow is connected to the inside or the perimeter of the element.
 - The endpoint is connected to a requirement, a requirement group, and the inside or the perimeter of the element.

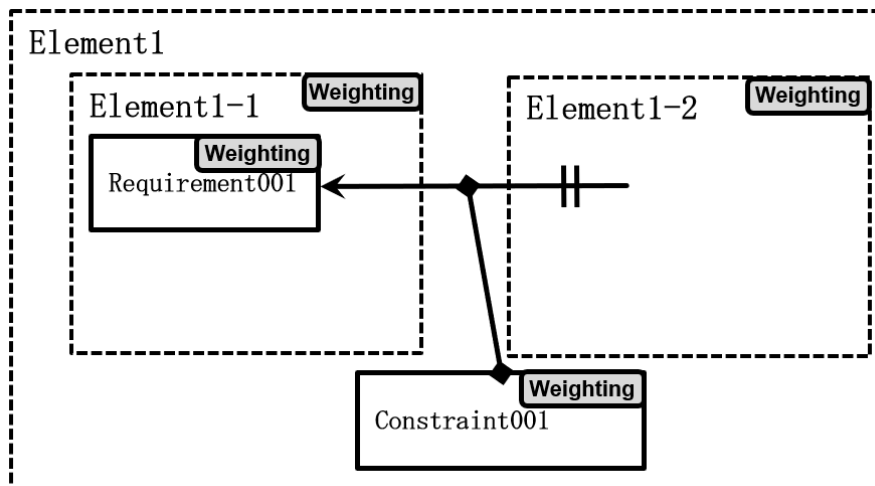


Figure 41 Example of notation for “freedom from interference” between elements and requirements

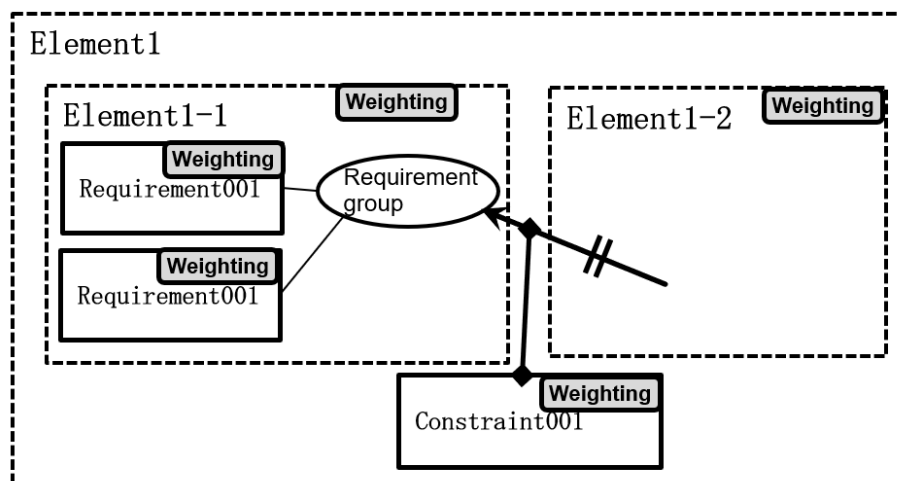


Figure 42 Example of notation for “freedom from interference” between elements and requirement groups

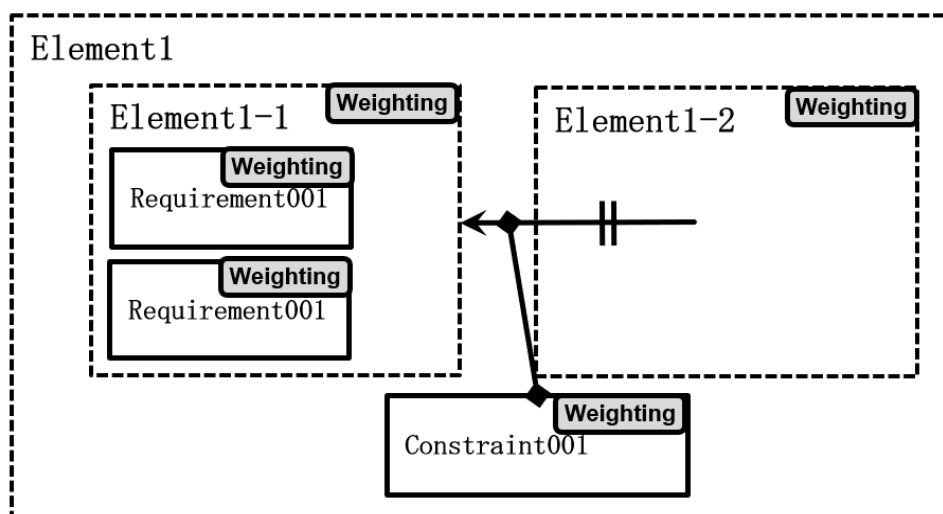


Figure 43 Example of notation for “freedom from interference” between elements

6.3.6 Branching of an interaction line between requirements

The following are notes for interaction lines between requirements after being allocated to the elements. There are no differences between the right figure and the left figure in Figure 44 in terms of the position of the branch point of the interaction lines between the requirements after being allocated to the elements. The branch point may be described by a blank circle as shown in the right and left figures in Figure 45.

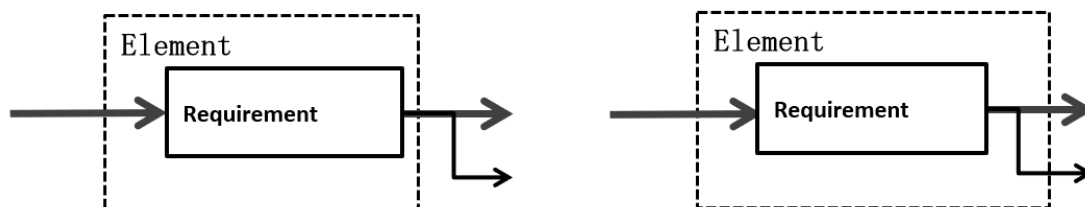


Figure 44 Branch point without any circles

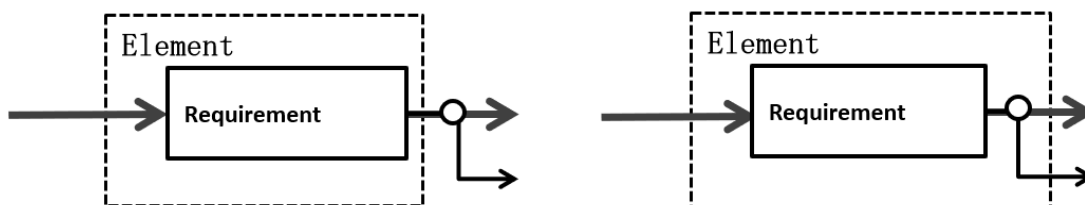


Figure 45 Branch point with a blank circle

To consider safety architecture, when describing allocation of the branch point on interaction lines between requirements to the element, “filled-in circles” are placed on the interaction lines to show the branch points. If branching outside the element, describe it as shown in the left figure of Figure 46. If branching inside the element, describe it as shown in the right figure of Figure 46.

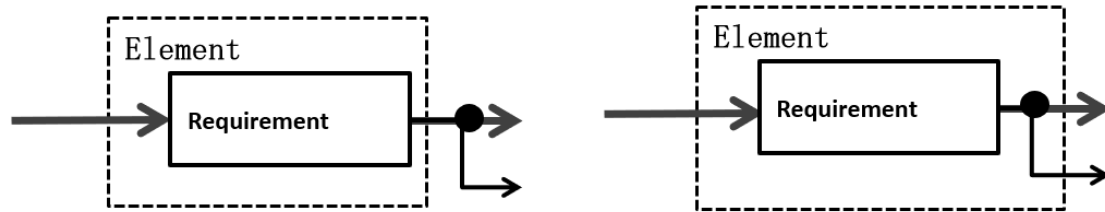


Figure 46 **Branch point with a filled-in circle**

Normative items

- Basic
 - The position of the branch point of interaction lines between requirements after being allocated to elements does not have any meaning until the branch point is described.
- Position and shape
 - The branch point may be described by a blank circle at intersections of lines. (See Figure 45)
 - When the allocation of the branch point on interaction lines between requirements to the element is described, “filled-in circles” are used to show the branch points. (See Figure 46)

7 Bibliography

ISO 26262:2018. Road Vehicles -- Functional Safety.

Appendix: A. SCDL metamodel

A.1. Overview

In this chapter, the SCDL metamodel is explained.

Figure A- 1 shows basic hierarchical relationships among the constituents of SCDL metamodel (hereinafter referred to as metamodel constituents). Note that attributes of metamodel constituents other than SCDL-type constituents are not described.

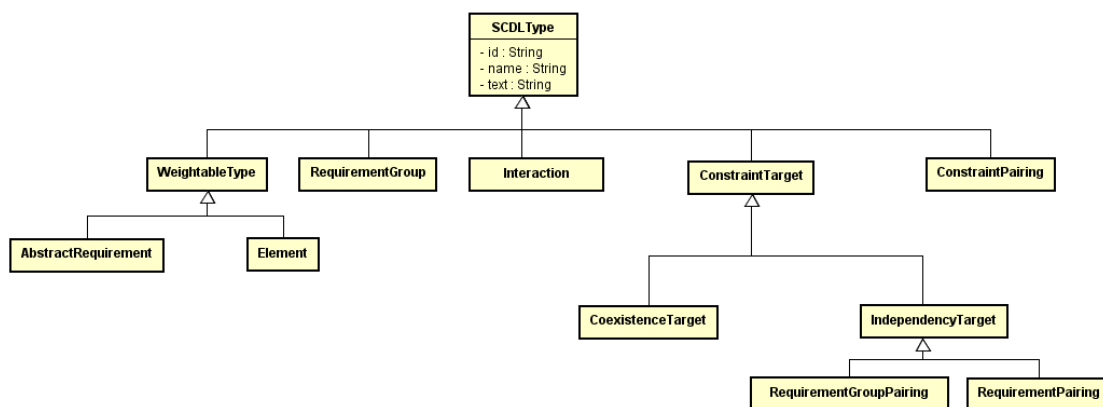


Figure A- 1 SCDL metamodel structure

Figure A- 2 shows relationships among metamodel constituents of SCDL and their attributes. However, SCDL-type constituents were omitted from this figure, as it would make this figure too complicated.

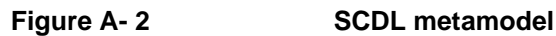


Table A- 1 Relationships between the specification and metamodel

ASAM SCDL Version 1.6.0

A.2. Scope

The scope of the SCDL model defined in this chapter is shown below.

- The basic definition of SCDL
 - Requirement
 - Requirement group
 - Element
 - Interaction
 - System boundary interaction
 - Constraint
 - Requirement group pairing
 - Connecting line from constraints
 - Freedom from interference
 - Weighting

A.3. SCDLType

“SCDL Type” is an abstract concept which becomes the basis for constituents with “id” or “name” among all constituents or relationships defined in SCDL.

A.3.1. Specializations

WeightableType, RequirementGroup, Interaction, ConstraintTarget, ConstraintPairing

A.3.2. Attributes

- id : String
Identifier
- name : String
Name, or abbreviated name
- text : String
Notes

A.3.3. WeightableType

“WeightableType” is an abstract concept showing constituents which can retain information of weighting.

A.3.4. Generalizations

SCDLType

A.3.5. Specializations

Element, AbstractRequirement

A.3.6. Association Ends

- weight: Weight[0..1]

A.4. Weighting

Constituent: “Weighting”. For its positioning, see 5.2 of this document.

A.5. AbstractRequirement

“AbstractRequirement” is an abstract concept showing the common parts between “Requirement” and “Constraint”.

A.5.1. Generalizations

WeightableType

A.5.2. Constraints

- If “isAllocated” is “true”, “allocation” exists.
- If “isAllocated” is “false”, “allocation” does not exist.

A.5.3. Specializations

Requirement, Constraint

A.5.4. Attributes

- isAllocated: boolean = false

If this is “false”, it represents that the allocation destination of a requirement to an element has not been decided yet. If this is “true”, a requirement has already been allocated to an element.

A.5.5. Association Ends

- allocation : Element[0..1]

Elements as allocation destination for requirements

A.6. Element

“Element” is a constituent which corresponds to elements described in 6.2.5.

A.6.1. Generalizations

WeightableType, InterferenceTarget

A.6.2. Constraints

Constituent “Element” of its own shall not be included in elements, or in elements of Element which is recursively included in the elements.

A.6.3. Association Ends

- parent: Element[0..1]
Parent element of an element
- elements: Element[0..*]
An element included in element

A.7. Requirement

“Requirement” is a constituent which corresponds to requirements described in 6.2.1.

A.7.1. Generalizations

AbstractRequirement, InterferenceTarget

A.7.2. Association Ends

- /incoming : Interaction[0..*]
Incoming interaction to a requirement
- /outgoing : Interaction[0..1]
Outgoing interaction from a requirement
- /group : RequirementGroup[0..*]
Group which a requirement belongs to

A.8. Constraint

“Constraint” is a constituent which corresponds to constraints described in 6.3.3.

A.8.1. Generalizations

AbstractRequirement

A.8.2. Association Ends

- pairing : ConstraintPairing[1..*]
This represents targets of constraints.

A.9. ConstraintPairing (Relation with constraints)

“ConstraintPairing” is an abstract concept showing relations between pairing and constraints defined in 6.3.3, or between freedom from interference and constraints defined in 6.3.5.

A.9.1. Generalizations

SCDLType

A.9.2. Association Ends

- constraint : Constraint[1..1]
Related constraints
- constraintTarget : ConstraintTarget[1..1]

Targets related to constraints

A.10. Interaction

“Interaction” is a constituent which corresponds to interactions described in 6.2.2.

A.10.1. Generalizations

SCDLType

A.10.2. Constraints

- Constituents “source” and “target” shall not be the same Requirement constituents.

A.10.3. Association Ends

- source : Requirement [1..1]
A requirement on the input/From the side of an interaction
- target : Requirement [1..*]
A requirement on the output/To the side of an interaction

A.11. RequirementGroup

“RequirementGroup” is a constituent which corresponds to requirement groups described in 6.3.2.1.

A.11.1. Generalizations

SCDLType, InterferenceTarget

A.11.2. Constraints

- Requirement constituents which are included in “requirements” shall not overlap.

A.11.3. Association Ends

- requirements : Requirement[1..*]
A requirement that which belongs to a requirement group

A.12. ConstraintTarget (Target to which constraints are applied)

“ConstraintTarget” is an abstract concept showing targets to which a constraint is applied.

A.12.1. Generalizations

SCDLType

A.12.2. Specializations

CoexistenceTarget, IndependencyTarget

A.12.3. Association Ends

- pairing: ConstraintPairing [1..*]

A.13. IndependencyTarget

“IndependencyTarget” is an abstract concept showing relations between pairing and constraints defined in 6.3.3.

A.13.1. Generalizations

ConstraintTarget

A.13.2. Specializations

RequirementGroupPairing, RequirementPairing

A.14. CoexistenceTarget

CoexistenceTarget is a constituent which corresponds to freedom-from-interference relevance described in 6.3.5.

A.14.1. Generalizations

ConstraintTarget

A.14.2. Association Ends

- target : InterferenceTarget [1..1]
Represents constituents of interference target
- source : Element[1..1]
Represents constituents of the interference source

A.15. InterferenceTarget

“InterferenceTarget” is an abstract concept showing freedom-from-interference-relevant interference target.

A.15.1. Specializations

Requirement, RequirementGroup, Element

A.16. RequirementGroupPairing

“RequirementGroupPairing” is a constituent which corresponds to the requirement group pairing described in 6.3.2.2.

A.16.1. Generalizations

IndependencyTarget

A.16.2. Constraints

Two “RequirementGroup” constituents of “set” shall be different constituents.

A.16.3. Association Ends

- set : RequirementGroup[2..2]
Two requirement groups in pairing relationships

A.17. RequirementPairing (Partial pairing)

“RequirementPairing” is a constituent which corresponds to the partial relationships between requirements which were detailed from requirement group pairing described in 6.3.4.

A.17.1. Generalizations

IndependencyTarget

A.17.2. Constraints

Two “Requirement” constituents of “set” shall be different constituents.

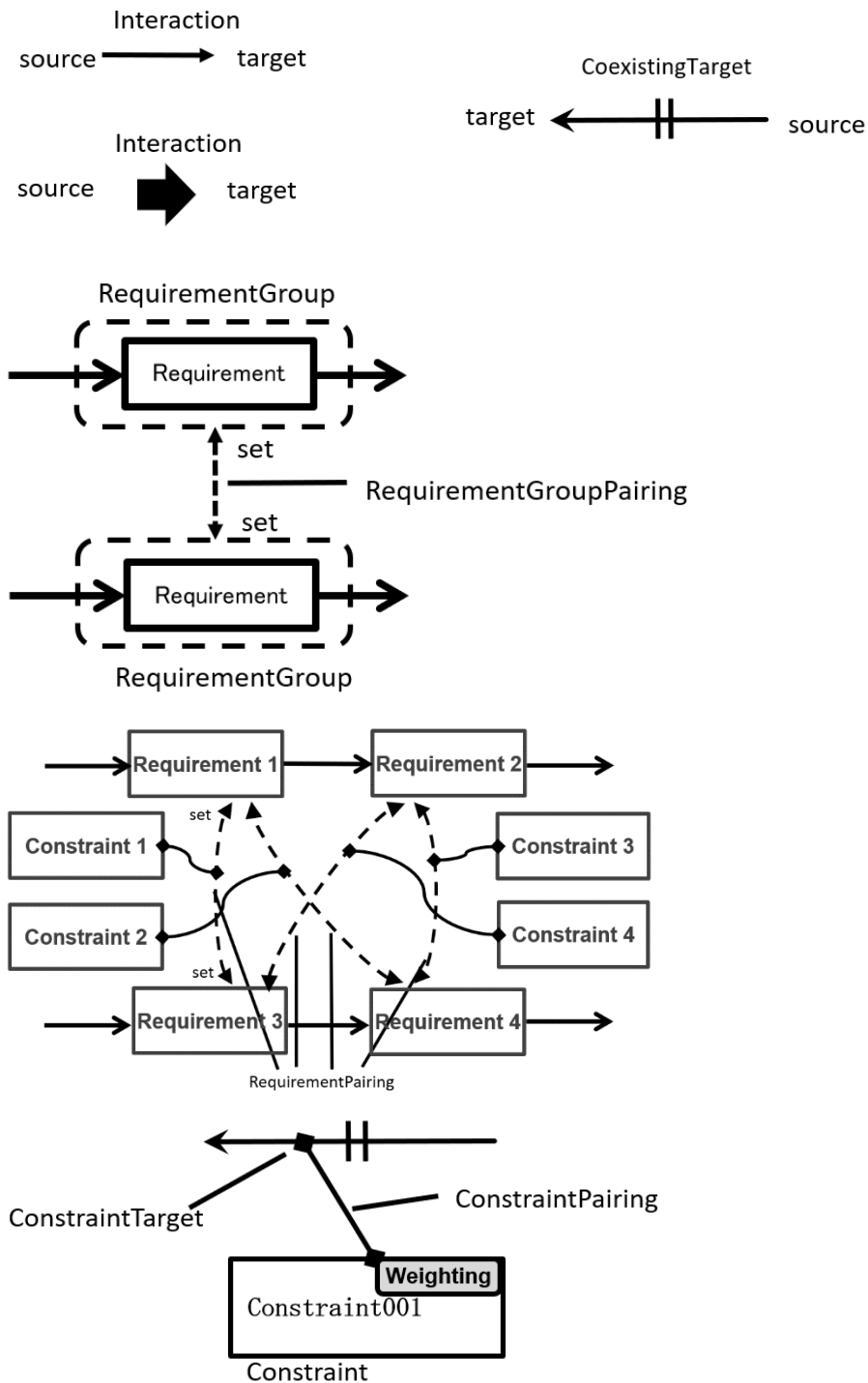
A.17.3. Association Ends

- set : Requirement[2..2]
Two requirements in independent condition relationships

A.18. Relationships between SCDL metamodel and figures

This chapter explains relationships between metamodel constituents and some figures described in this specification to facilitate understanding of the metamodel.

A.18.1. Relationships in figures in the specification



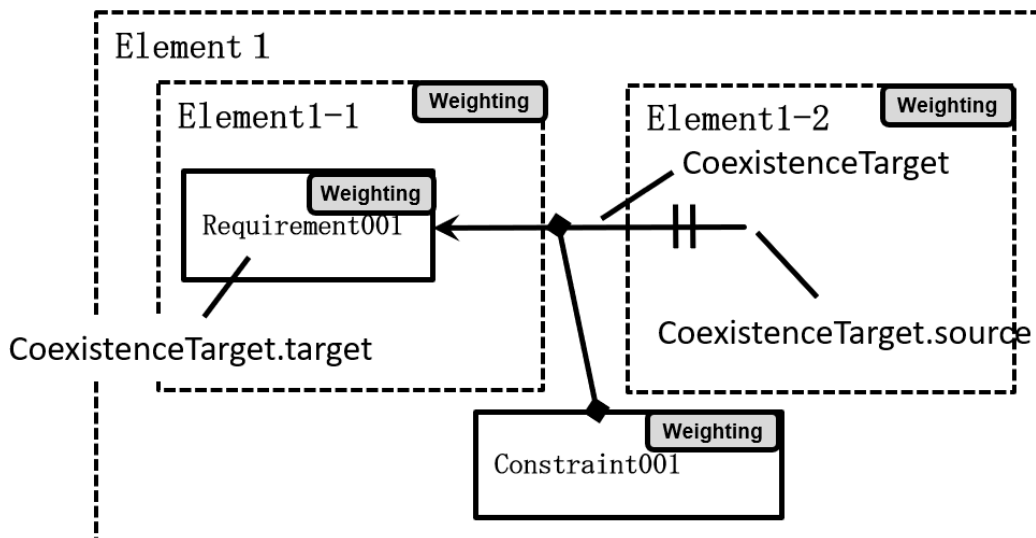


Figure A-3 Relationships between metamodel and figures (notation)

A.18.2. Relationships considering figures used for use case examples

Relationships with metamodel constituents:

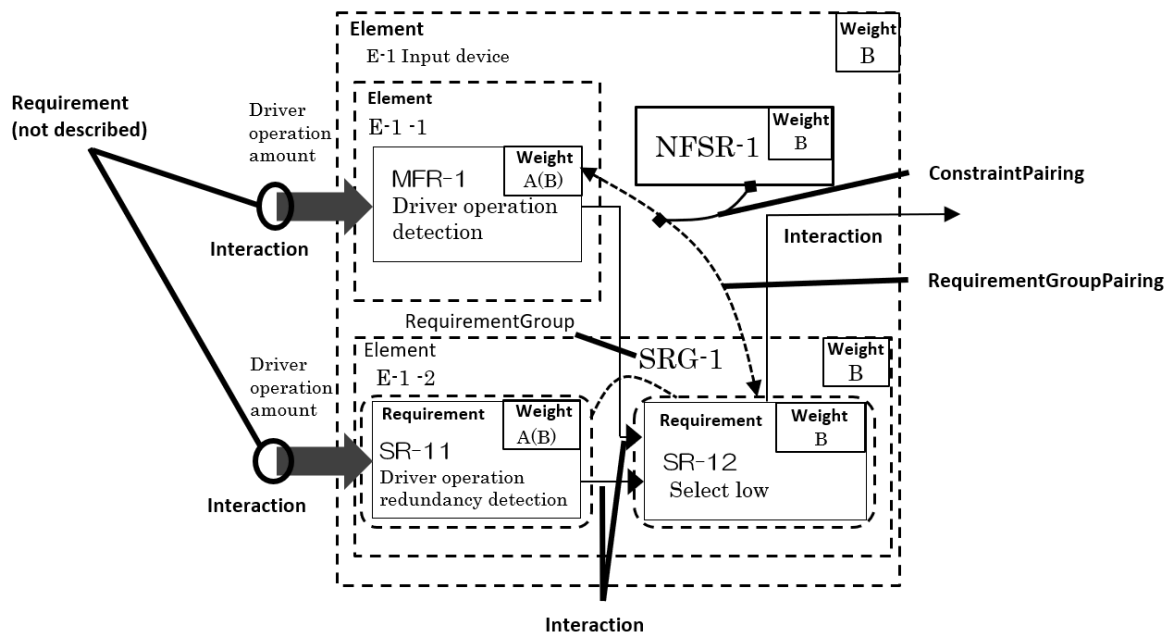


Figure A-4 Relationships between metamodel and use case examples

Relationships with attributes of metamodel constituents:

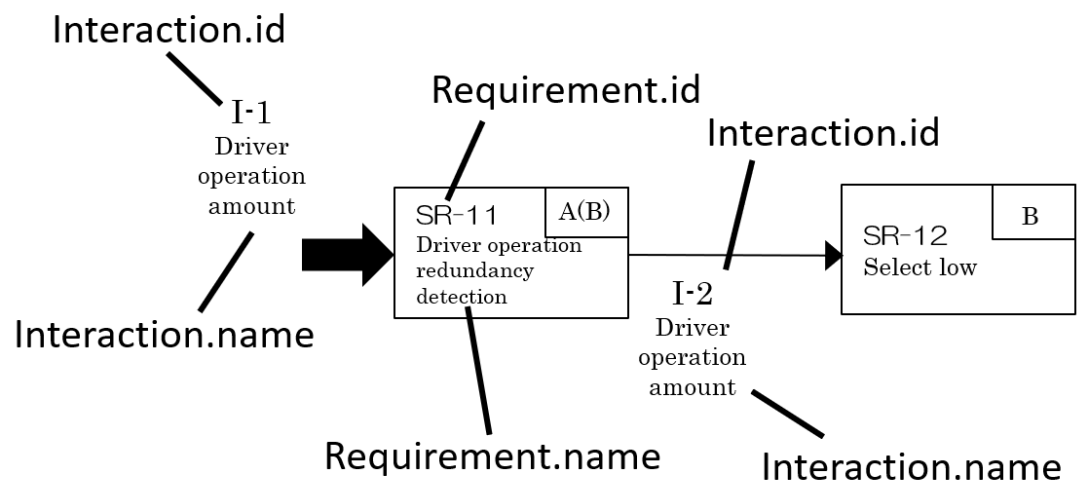


Figure A- 5 Relationships with attributes of metamodel constituents

Figure Directory

Figure 1	Specification of safety mechanism	12
Figure 2	Identification and separation of safety-related portions	13
Figure 3	Notation of requirements.....	18
Figure 4	Example of the case with multiple input interactions	19
Figure 5	Example of branching	19
Figure 6	Representative notation of requirements and interactions between requirements.....	20
Figure 7	Example of notation for system boundary interactions	20
Figure 8	External plant.....	21
Figure 9	Example of use of external plant	21
Figure 10	Notation for Other technology links	22
Figure 11	Example of use of Other technology links	23
Figure 12	Prohibition: Avoid two or more output interactions from one requirement....	24
Figure 13	Prohibition: Input interactions shall not be merged.....	24
Figure 14	Example of a requirement without inputs	24
Figure 15	Example of a requirement without outputs	24
Figure 16	Notation for elements.....	25
Figure 17	Example of prohibition when describing elements (division).....	26
Figure 18	Example of prohibition when describing elements (overlap).....	26
Figure 19	Example of prohibition when describing elements (inscribing)	26
Figure 20	Notation of constraints when the destination in an element has not been decided	27
Figure 21	Notation of connecting line.....	28
Figure 22	Example of a connecting line connected with freedom from interference	28
Figure 23	Description of allocation of requirement and element.....	29
Figure 24	Assignment of weighting for element	29
Figure 25	Example of prohibition of inscribing.....	30
Figure 26	Example of prohibition of intersecting.....	30
Figure 27	Example of adequate requirement allocation to the nested element	30
Figure 28	Notation for framed requirement group	31
Figure 29	Notation for connected requirement group	32
Figure 30	Notation for “Balloon-type” requirement group	32
Figure 31	Notation for “Tab-type” requirement group	33
Figure 32	Combination of framed requirement groups	34
Figure 33	Combination of connected requirement groups.....	35
Figure 34	Notation for “Balloon-type” requirement group	36
Figure 35	Notation for “Tab-type” requirement group pairing.....	37
Figure 36	Notation for constraints for pairing	38
Figure 37	Constraints	38
Figure 38	Notation for constraints by “Balloon type”.....	39
Figure 39	Notation for constraints by “Tab type”	40
Figure 40	Detailing constraints.....	41
Figure 41	Example of notation for “freedom from interference” between elements and requirements.....	42
Figure 42	Example of notation for “freedom from interference” between elements and requirement groups.....	42
Figure 43	Example of notation for “freedom from interference” between elements.....	43
Figure 44	Branch point without any circles.....	43
Figure 45	Branch point with a blank circle.....	43
Figure 46	Branch point with a filled-in circle	44

Figure A- 1	SCDL metamodel structure	46
Figure A- 2	SCDL metamodel.....	47
Figure A- 3	Relationships between metamodel and figures (notation)	55
Figure A- 4	Relationships between metamodel and use case examples	55
Figure A- 5	Relationships with attributes of metamodel constituents	56

Table Directory

Table 1	Scope of SCDL	14
Table 2	SCDL basic definition (constituents)	16
Table 3	SCDL basic definition (relationship)	17
Table A- 1	Relationships between the specification and metamodel	47



Association for Standardization of
Automation and Measuring Systems

E-mail: support@asam.net

Web: www.asam.net

© by ASAM e.V., 2021