

Writing requirements

**Date:** 2013.12.16

**Project/Technology area:** Functional Safety

**Phase:** N/A

**Document:** Writing Requirements T-13-0158.docx

Table of Contents

[1 Scope 2](#_Toc374976222)

[1.1 Basic Structure of a Requirement 2](#_Toc374976223)

[1.2 Complete Requirements 3](#_Toc374976224)

[1.3 Consistent Requirements 3](#_Toc374976225)

[1.4 Atomic Requirements 3](#_Toc374976226)

[1.5 Unambiguous Requirements 3](#_Toc374976227)

[1.6 Verifiable Requirements 4](#_Toc374976228)

[1.7 Traceable Requirements 4](#_Toc374976229)

[2 Document history 4](#_Toc374976230)

# Scope

The organization shall ensure that the persons involved in the execution of the safety lifecycle have a sufficient level of skills, competences and qualifications corresponding to their responsibilities. This document defines the objectives for writing “good” functional requirements or technical requirements. The source of this information is partly inspired from [1] and [2].

## Basic Structure of a Requirement

This following is a basic structure of a statement that is clear and concise in explaining the intention:

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** (the entity doing something) | **Strong command** (shall, must, will, shall not, cannot, must not) | **Main Verb** (what is to be done) | **Target** (the target of the actor) |
| The WIPER CONTROL FUNCTION | shall | activate | the Front Wiper Low Speed. |

“The WIPER CONTROL FUNCTION shall activate the Front Wiper Low Speed”.

Conditional statements are then added to steer the activation:

|  |  |  |
| --- | --- | --- |
| **Switch** | **Object** | **Condition** |
| IF | the Low Speed Switch | Is detected as Active |

“IF the Low Speed Switch is detected as Active”.

Statement + Condition = Condition + Statement = Requirement.

“The WIPER CONTROL FUNCTION shall activate the Front Wiper Low Speed IF the Low Speed Switch is detected as Active”.  
“IF the Low Speed Switch is detected as Active, the WIPER CONTROL FUNCTION shall activate the Front Wiper Low Speed”.  
Both requirements express the same thing and are equally valid.

Avoid using weak commands such as “may”, “can”, “possibly can”, “can try to”. It implies additional (hidden) conditions that must be fulfilled, and leaves the reader with a question what is meant. Be concise!

## Complete Requirements

A complete requirements specification must precisely define all the real world situations that will be encountered. If conditional statements are used, it must be specified what to do for every possible condition. Leaving open ends can lead to implementation errors.

“The WIPER CONTROL FUNCTION shall activate the Front Wiper Low Speed IF the Low Speed Switch is detected as Active” does not explain what is intended if the Low Speed Switch is detected an Inactive, or not even detected at all. It leaves open questions when doing an implementation.

“The WIPER CONTROL FUNCTION shall activate the Front Wiper Low Speed IF the Low Speed Switch is detected as Active”, ELSE “The WIPER CONTROL FUNCTION shall de-activate the Front Wiper Low Speed”.

Adding the ELSE statement defines the remainder of the real world situations that will be encountered.

## Consistent Requirements

A consistent specification has no conflict between individual requirements. Inconsistencies must be either eliminated by changing the original requirement, or by adding statements to solve the inconsistency.

“The WIPER CONTROL FUNCTION shall activate the Front Wiper Low Speed IF the Low Speed Switch is detected as Active”, ELSE “The WIPER CONTROL FUNCTION shall de-activate the Front Wiper Low Speed”.

“The WIPER CONTROL FUNCTION shall activate the Front Wiper High Speed IF the Low Speed Switch and the High Speed Switch is detected as Active”, ELSE “The WIPER CONTROL FUNCTION shall de-activate the Front Wiper High Speed”.

These two requirements are clearly in conflict, because you cannot have Low Speed and High Speed active at the same time. Implementing these requirements needs a definition of the priority between the two wiper functions, which is missing in the requirements.

“The Front Wiper High Speed has priority over Front Wiper Low Speed” solves the inconsistency.

## Atomic Requirements

Multiple nested requirements are difficult to read and understand, and opens up for implementation errors. Split up nested requirements, down to the basic form in section 1.1.

## Unambiguous Requirements

A statement of a requirement is unambiguous if it can only be interpreted one way. The use of weak phrases or poor sentence structure will open the specification statement to misunderstandings. Again, simplify down to the basic form in section 1.1 to assist a clear interpretation.

To support the addition of background information to a requirement, or directions on testing, without mixing this information with the requirement itself, it is strongly recommended to use the requirements template as defined in System Weaver; **Background** / **Requirement** / **TestHints**.

## Verifiable Requirements

In order to be verifiable, requirement must be linked to attributes possible to judge as pass/fail. Avoid weak words when defining attributes.

“The Front Wiper Low Speed shall be sufficiently lower than the Front Wiper High Speed” is not a verifiable requirement. The word “sufficiently” is weak and lacks an attribute.

“The Front Wiper Low Speed shall be 50% +/- 6% of the Front Wiper High Speed” is a verifiable requirement.

Note that subjective judgment can be assigned attributes if there is a subjective scale to measure against.

## Traceable Requirements

Requirements shall be tagged or numbered so that they are traceable upwards to their origin, and downwards to the next lower abstraction level. Software based tools such as Doors or System Weaver support this natively.

# Document history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Update** | **Reference** | **Who** | **Date** | **Version** |
| Initial release | Safety Engineering Process | S Vertetics | 2013-12-16 | 1.0 |

Appendix A - References

The following documents are either referenced by this document or contain related information.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Document name** | **Version** | **Document title** |
| [1] | Web link | --- | <http://en.wikiversity.org/wiki/Technical_writing_specification> |
| [2] | Article in The Journal of Defense Software Engineering | --- | Writing Effective Natural Language Requirements Specifications, William M. Wilson, 1999 |