White Paper

VISION:EER

How to handle Generic Requirements ?

Req1: **The application must be the**

How to handle Generic Requirements?

1. What are **Generic Requirements**?

The following Generic Requirements exist in any project :

- Development Goals (*e.g. "economic system")*
- Standards or Regulations (*e.g. Safety Rules,..)*
- Legal Requirements
- Security and IT Requirements
- Stakeholder or Component Requirements

Generic requirements:

- **Are vague requirements**

→ *They must be linked with a certain bundle of derived concrete-requirements (solutions)*

- **Are assigned to a certain entity** (*e.g. a Vehicle*)

→ *They must be reused by each member of the assigned entity separately*

- **Must be specialzed for each derived sub-entity** (e.g. a Pic Up)

→ *All common requirements of each sub-entity must be added to the assigned solutions separately*

2 . What is the **Consequence**?

- According to the **Single Source of Truth** (SSOT), these requirements may only be defined in **one location**.

  However, duplicates are unavoidable for describing the missing **commonalities** and **details** of each **subentity-group** and the specific requirements of each **individual group member** (instance).

  ➔ **Automatic synchronization of the reused items required**

- It must contain all information regarding the **reuse rules** (*mandatory, erasable, overwritable, proposal etc.)*

  ➔ **Automatic verification required, if the reuse rules are followed**

- Generic Requirements are **incomplete** and must be specialized and detailed (extended) on each subentity level by different organizations, but must be finally **completed** on lowest etity-level (specifcation instance)

  ➔ **Automatic verification required, if each instance is described completely**

Note: This is **only achievable** with

object-orientent methods and **tool support!**

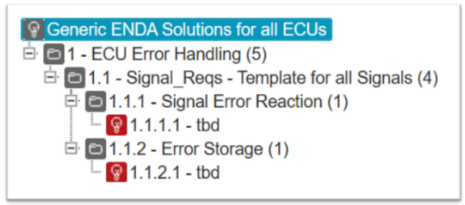
3. Example: **Legal Requirements**

In the following **simplified** example, an **ENDA** (emission neutral default action) **law** is defined in a database for the applicable laws within an automotive-company and is linked to a bundle of solutions, which are defined in a solutions-library:

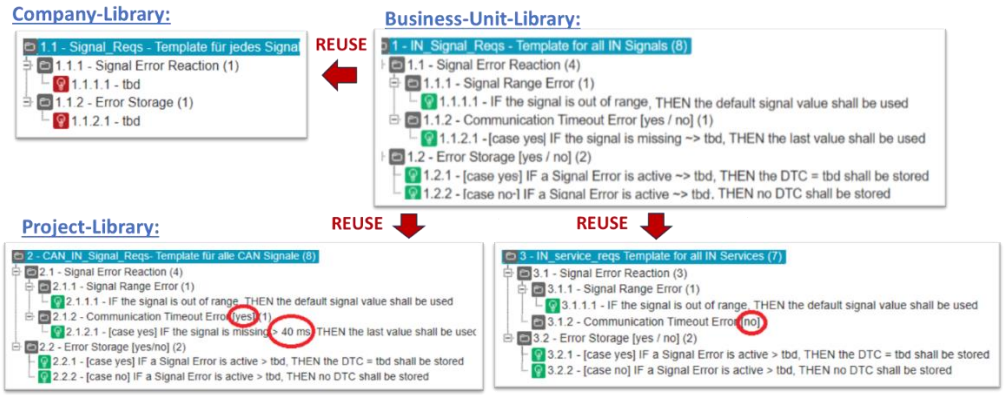
Database for applicable laws:

linked

Solutions-Library:



As depicted in the following diagram, a linked solution is e.g. a generic template for all signals, which secures that for each of the signals the Signal Error Reaction and the Error Storage are specified (to ensure that it is implemented and tested), without defining the details how this shall be performed (by using the placeholder *tbd*).



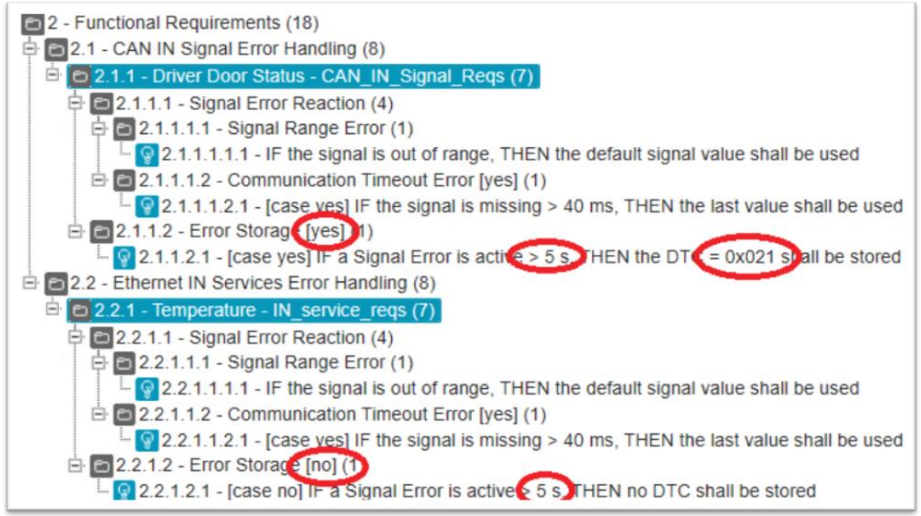
The solution-template, which is in this example valid for the whole company, is then reused and specialized by various suborganizations. As depicted, each reusing organization is defining the common requirements for specific entities, which shall be reused by its suborganizations.

In this example, a business unit has defined a derived template for all **Input signals** with details for the error reaction, e.g. what tests shall be performed and the signal replacement strategy in case of a faulty signal. Additionally,.case relevant requirements are defined, which are obligatory for the reuse, but its specific decisions shall be taken by suborganizations.

On project level this example template is reused to create specialized templates that contain the common requirements for CAN signals and for Services, in which commonalities (specific case decisions and values) of these **signal-types** are fixed.

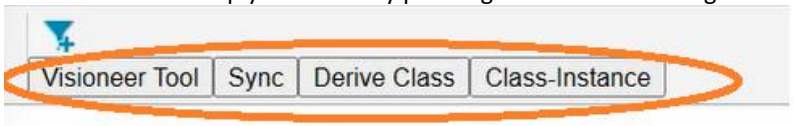
These signal-type specific templates shall be reused e.g. when creating an ECU-Specification, in which the error handling for each ECU-signal shall be described (for each signal separately). Finally, in the **specification** the missing **signal specific** values and case decisions must be added, as shown in the following example:

ECU Specification:



4. Solution: **Visioneer Tool**

The Visioneer Tool is an AddOn (currently only available) for Codebeamer, which handles textual requirement templates in **classes** with **object-oriented methods.** The tool-functions can be simply executed by pressing one of the following buttons:



The examples tracker "Template Klassen" contains the parent-class *IN_Signal_Reqs* and the derived child-classes *CAN_IN_Signal_Reqs* and *IN_service_reqs,,* which **inherit** all items from their **parents** and contain all commonalities of these specific signal-types:

- Template Klassen
  - 1 - IN_Signal_Reqs (8)
    - 1.1 - Signal Error Reaction (4)
      - 1.1.1 - Signal Range Error (1)
        - 1.1.1.1 - IF the signal is out of range~, THEN the last signal value shall be used
      - 1.1.2 - Communication Timeout Error [yes / no] (1)
        - 1.1.2.1 - ~[case yes~] IF the signal is missing ~> tbd~, THEN the default value shall be used
    - 1.2 - Error Storage [yes / no] (2)
      - 1.2.1 - ~[case yes~] IF a Signal Error is active ~> tbd~, THEN the DTC = tbd shall be stored
      - 1.2.2 - ~[case no~] IF a Signal Error is active ~> tbd~, THEN no DTC shall be stored
  - 2 - CAN_IN_Signal_Reqs extends IN_Signal_Reqs (8)
    - 2.1 - Signal Error Reaction (4)
      - 2.1.1 - Signal Range Error (1)
        - 2.1.1.1 - IF the signal is out of range~, THEN the last signal value shall be used
      - 2.1.2 - Communication Timeout Error [yes] (1)
        - 2.1.2.1 - [case yes] IF the signal is missing > tbd, THEN the default value shall be used
    - 2.2 - Error Storage [yes / no] (2)
      - 2.2.1 - [case yes] IF a Signal Error is active > 40 ms, THEN the DTC = tbd shall be stored
      - 2.2.2 - [case no] IF a Signal Error is active > tbd, THEN no DTC shall be stored
  - 3 - IN_service_reqs extends IN_Signal_Reqs (8)
    - 3.1 - Signal Error Reaction (4)
      - 3.1.1 - Signal Range Error (1)
        - 3.1.1.1 - IF the signal is out of range, THEN the last signal value shall be used
      - 3.1.2 - Communication Timeout Error [yes / no] (1)
        - 3.1.2.1 - [case yes] IF the signal is missing > tbd, THEN the default value shall be used
    - 3.2 - Error Storage [yes / no] (2)
      - 3.2.1 - [case yes] IF a Signal Error is active > tbd, THEN the DTC = tbd shall be stored
      - 3.2.2 - [case no] IF a Signal Error is active > tbd, THEN no DTC shall be stored

The tracker "ECU Spec" contains examples for a **controlled reuse** of these template classes in an ECU project specification:

- ECU Spec
  - 1 - Scope
  - 2 - Functional Requirements (20)
    - 2.1 - CAN IN Signal Error Handling (9)
      - 2.1.1 - Driver Door Status - CAN_IN_Signal_Reqs (8)
        - 2.1.1.1 - Signal Error Reaction (4)
          - 2.1.1.1.1 - Signal Range Error (1)
            - 2.1.1.1.1.1 - IF the signal is out of range, THEN the last signal value shall be used
          - 2.1.1.1.2 - Communication Timeout Error [yes] (1)
            - 2.1.1.1.2.1 - [case yes] IF the signal is missing > tbd, THEN the last value shall be used
        - 2.1.1.2 - Error Storage [yes / no] (2)
          - 2.1.1.2.1 - [case yes] IF a Signal Error is active > 40 ms, THEN the DTC = tbd shall be stored
          - 2.1.1.2.2 - [case no] IF a Signal Error is active > tbd, THEN no DTC shall be stored
    - 2.2 - Ethernet IN Services Error Handling (9)
      - 2.2.1 - Temperature - IN_Signal_Reqs (8)
        - 2.2.1.1 - Signal Error Reaction (4)
          - 2.2.1.1.1 - Signal Range Error (1)
            - 2.2.1.1.1.1 - IF the signal is out of range, THEN the last signal value shall be used
          - 2.2.1.1.2 - Communication Timeout Error [yes / no] (1)
            - 2.2.1.1.2.1 - [case yes] IF the signal is missing > tbd, THEN the default value shall be used
        - 2.2.1.2 - Error Storage [yes / no] (2)
          - 2.2.1.2.1 - [case yes] IF a Signal Error is active > tbd, THEN the DTC = tbd shall be stored
          - 2.2.1.2.2 - [case no] IF a Signal Error is active > tbd, THEN no DTC shall be stored
  - 3 - Non-Functional Requirements (1)
    - 3.1 - tbd

The Visioneer-Tool is performing:

- ✓ **Automatic synchronization** of the reused items, regarding parent-changes:
  - Adding requirements
  - Deletion of requirements
  - Modification of requirement structures
  - Modification of the text. descriptions
  - Modification of decision requirements, attribute field contents or links
- ✓ **Automatic verification**,
  - if the reuse rules are followed
  - if each instance is described completely

More details about it and how to get a **Demo Version** on our home page:

www.visioneer.info