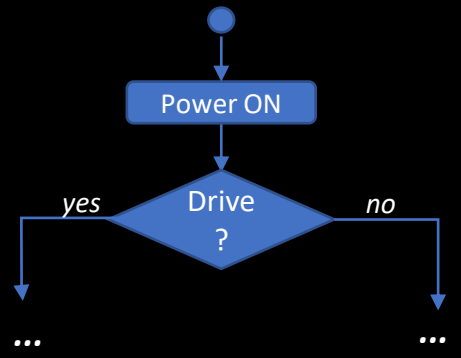
VISION:EER

www.visioneer.info

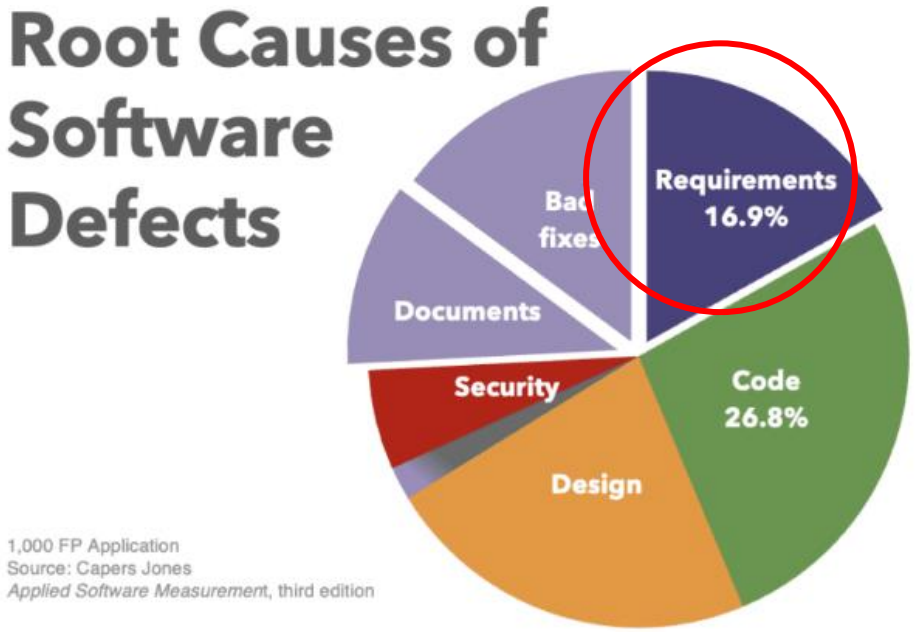
Requirements In-the-Loop (RIL) Tests

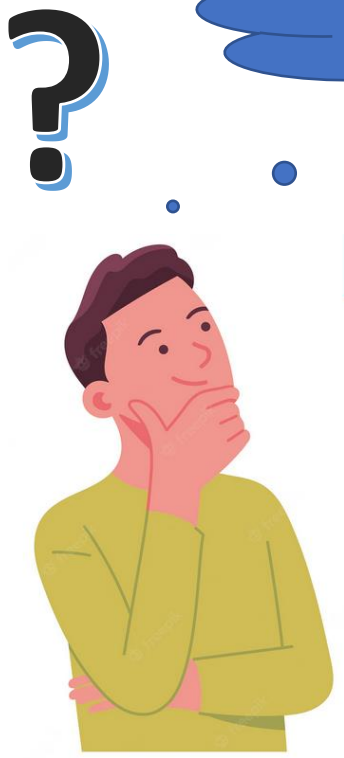


Contents

VISION:EER

Problemstellung





Wie vermeidet man <u>ungenügende</u> Requirement-Spezifikationen ?

<u>Erwartete Eigenschaften:</u>

- Notwendig
- Nachvollziehbar
- Korrekt
- Machbar
- Testbar

} Tool-support verfügbar

- Vollständig
- Konsistent
- Eindeutig
- Konform
- Einzigartig

} **Kein ausreichender Tool-support ***

**** Nur ca. 50% der erwarteten Eigenschaften können mit <u>herkömmlichen Tools</u> verifiziert werden!***

VISION:EER

Idee

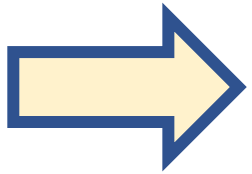
" Durch eine Standardisierung von textuellen Anforderungselementen:

→ Alle erwarteten Eigenschaften können automatisch verifiziert werden"

Standardisierungs-Maßnahmen:

- Standardisierung von **Spezifikationsinhalten** und **Syntax**
- **Klassifizierung** der funktionalen Anforderungen
- Wiederverwendung von Spezifikationselementen durch **Template-Klassen**
- Klassen können abgeleitet oder instanziiert und **Reqs** und **Spezifikationsstrukturen** somit **vererbt** werden

Automatische RIL Tests



Die fehlenden Überprüfungen können durchgeführt werden:

- ***Vollständigkeit***
- ***Konsistenz***
- ***Eindeutigkeit***
- ***Konformität***
- ***Einzigartigkeit***

Test Protokol

VISION:EER

Lösung

Visioneer bietet folgende Dienstleistungen/Produkte an:

1. **Entwicklung eines RE-Prozesses zur Erzielung der Req-Eigenschaften**

   → **RE-Consulting in Partnerschaft mit:**
   - **Engineering People GmbH (EU)**
   - **SMSTT Ltd (USA und Asien)**

2. **Zur Durchführung der fehlenden RIL-Tests**

   → **Visioneer-Tool**
   - **AddOn für RE-Tools (CodeBeamer, Doors..)**
   - **Handling von text. Reqs in Klassen**
   - **RIL-Test Automatisierung**

VISION:EER

Visioneer RE-Prozess

Methode:

1. Verwendung der beschriebenen Standardisierungsmaßnahmen

2. Entwicklung eines RE-Prozesses
   - Definition der Kriterien für die Erfüllung der erwarteten Anforderungen
   - Definition der Maßnahmen, die zur Erfüllung der einzelnen Kriterien erforderlich sind
   - Erstellung von Automatismen zur Generierung der Nachweise

3. Beschreiben von wiederverwendbaren Reqs-Strukturen in Klassen

VISION:EER

Beispiel RE-Prozess

Vollständigkeits-kriterien	Maßnahmen	Nachweis
Funktionale Vollständigkeit	• Jede Stakeholder-Anforderung muss mit einer Systemanforderung verknüpft werden • ...	herkömmliches Traceability-Tool
Logische Vollständigkeit	• Alle logischen Eingangssignalkombinationen müssen klar definierte Ausgangswerte	VISIONEER-Tool
.....		

Das VISIONEER-Tool ermöglicht die fehlenden 50% der RIL-Tests

VISION:EER

Fehlende RIL-Tests

Mit **herkömmlichen Tools** können keine Nachweise für die folgenden Kriterien erstellt werden:

Vollständigkeits-Kriterien	Maßnahmen
Logische Vollständigkeit	Es muss überprüft werden, ob alle pot. Kombinationen einen definierten Ausgangswert haben
Vollständigkeit der Details	Es ist zu überprüfen, ob alle vorgeschriebenen Template-Items Details wiederverwendet werden
Beschreibung der Fehlerbehandlung und der SUP-Funktionen	Es ist zu überprüfen, ob die Spezifikation alle Reqs-Strukturen (inkl. Fehlerbehandlungs- und Supplementfunktionen) enthält, welche in der entsprechenden Parent-Klasse (Baukasten) gefordert ist

VISION:EER

Fehlende RIL-Tests

Mit **herkömmlichen Tools** können keine Nachweise für die folgenden Kriterien erstellt werden:

Konformitäts-Kriterien	Maßnahmen
Die Anforderungs-Spezifikationsstruktur muss konform zu den Systemarchitektur-Elementen (inkl. Interfaces) sein	Es muss überprüft werden, Anforderungs-Spezifikationsstruktur konform zu den Systemarchitektur-Elementen (inkl. Interfaces) ist

VISION:EER

Fehlende RIL-Tests für el. Systeme

Eindeutigkeit

Es muss klar sein, welche LIB-Elemente **obligatorisch** oder **optional** sind und welche überschrieben oder gelöscht werden dürfen

Einzigartigkeit

Jede Req wird nach der **Single-Source-of-Truth** gehandhabt

Konsistenz

Jedes Mitglied einer Entity muss mit den **gleichen Spezifikationsstruktur-Elementen** beschrieben werden

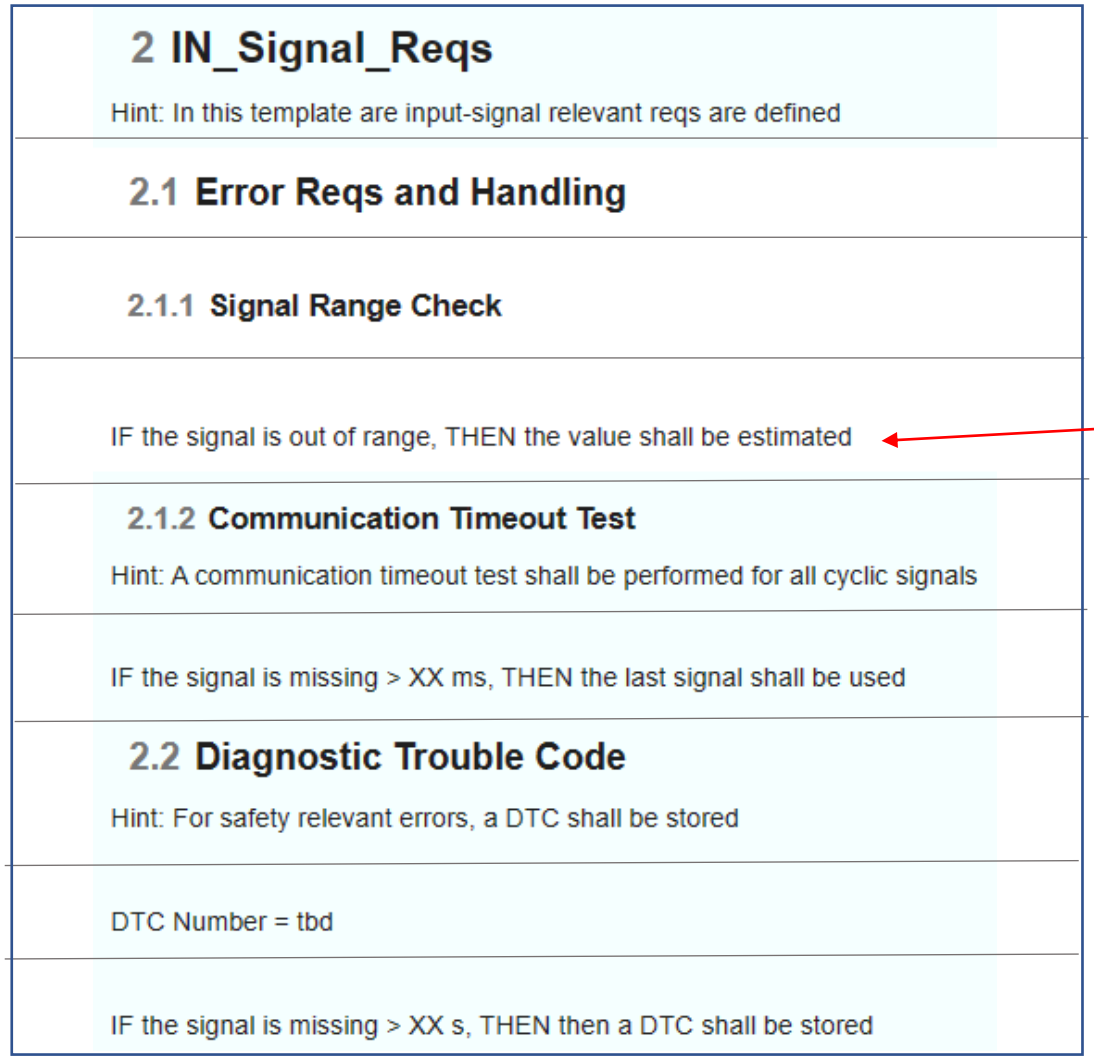
Maßnahmen

Die Wiederverwendung von Templates soll mit **objektorientierten Methoden (Klassen)** erfolgen

→ Es ist zu überprüfen, ob die Vererbungsregeln eingehalten werden

VISION:EER

Derzeitige Req Templates



In dieser (vereinfachten) Beispiel-Template werden generische Anforderungen definiert

- welche für jedes IN-Signal relevant sind

→ **Generelles Wiederverwendungs-Probleme:**

- Einige Anforderungen sind vage und daher nicht testbar
- Es ist unklar, welche Anforderungen verpflichtend oder optional und was Vorschläge sind
- Es ist unklar, welche Anforderungen für welche Entity gemeinsam sind (z. B. für alle CAN-Signale)
- Es wird nicht automatisch überprüft, ob die Spezifikationen alle obligatorischen Elemente des Templates enthalten
-

→ **Hoher Review-Aufwand und Risiko für falsche oder fehlende Anforderungen**

VISION:EER

Visioneer Tool Version 1:

Requirement Template Classes

Lösung Requirement Template-Klassen

Mit Template-Klassen können textuelle Anforderungen mit objekt-orientierten Methoden vererbt werden:

- Es wird über ein Field definiert , ob eine Anforderung **obligatorisch, optional** oder ein **Vorschlag** ist bzw. ob es **überschrieben** oder **gelöscht** werden darf
- **Abgeleitete Klassen** enthalten alle Items der Parent-Klasse und alle gemeinsamen Reqs einer Entity
- **Klasseninstanzen** werden wiederverwendet, um die Reqs jedes einzelnen Mitgliedes einer Entity zu definieren (z. B. für jedes Eingangssignal)
- **Kardinalitäten** (z.B. 1..*) für die erlaubten Multiplizitäten von Req-Items können definiert werden
- Die Anforderungen werden in verschiedene **Typen** (z. B. Verhaltensanforderungen, Ziele) **klassifiziert**, welche separat verifiziert werden können
- Nur **case-spefische Reqs** werden **vererbt**

→ **AddOn für RE Tool, d.h. keine Synchronisation mit MBSE-Tool nötig**

VISION:EER

Lösung Requirement Template-Klassen

Durch das Visioneer-Tool werden abgeleiteten Klassen und Klasseninstanzen automatisch überprüft, ob diese den Vererbungsregeln ihrer Parent-Klassen folgen

VISION:EER

Beispiel Template-Klasse

	Type	Classification	Protection
1 IN_Signal_Reqs Hint: In this template are input-signal relevant reqs are defined	Class	-	-
1.1 Error Reqs and Handling	Folder	-	Mandatory
1.1.1 Signal Range Check	Folder	-	Mandatory
IF the signal is out of range, THEN the value shall be estimated	Functional Req	Functional Goal	Proposal
1.1.2 Communication Timeout Test [yes / no] Hint: A communication timeout test shall be performed for all cyclic signals	Folder	-	Mandatory
[case yes] IF the signal is missing > XX ms, THEN the last signal shall be used	Functional Req	Cond. Beh. Req.	Mandatory
1.2 Diagnostic Trouble Code [yes, no] Hint: For safety relevant errors, a DTC shall be stored	Folder	-	Mandatory
[case yes] DTC Number = tbd	Information	-	Mandatory
[case yes] IF the signal is missing > XX s, THEN then a DTC shall be stored	Functional Req	Cond. Beh. Req.	Mandatory

VISION:EER

Beispiel Template-Klasse

	Type	Classification	Protection
1 IN_Signal_Reqs Hint: In this template are input-signal relevant reqs are defined	Class	-	
1.1 Error Reqs and Handling	Folder	-	Mandatory
1.1.1 Signal Range Check	Folder	-	Mandatory
IF the signal is out of range, THEN the value shall be estimated	Functional Req	Functional Goal	Proposal
1.1.2 Communication Timeout Test [yes / no] Hint: A communication timeout test shall be performed for all cyclic signals	Folder	-	Mandatory
[case yes] IF the signal is missing > XX ms, THEN the last signal shall be used	Functional Req	Cond. Beh. Req.	Mandatory
1.2 Diagnostic Trouble Code [yes, no] Hint: For safety relevant errors, a DTC shall be stored	Folder	-	Mandatory
[case yes] DTC Number = tbd	Information	-	Mandatory
[case yes] IF the signal is missing > XX s, THEN then a DTC shall be stored	Functional Req	Cond. Beh. Req.	Mandatory

Element darf nicht in child gelöscht werden

Beispiel Template-Klasse

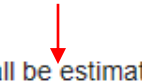
	Type	Classification	Protection
1 IN_Signal_Reqs Hint: In this template are input-signal relevant reqs are defined	Class	-	-
1.1 Error Reqs and Handling	Folder		landatory
1.1.1 Signal Range Check	Folder		landatory
IF the signal is out of range, THEN the value shall be estimated	Functional Req	Functional Goal	Proposal
1.1.2 Communication Timeout Test [yes / no] Hint: A communication timeout test shall be performed for all cyclic signals	Folder	-	Mandatory
[case yes] IF the signal is missing > XX ms, THEN the last signal shall be used	Functional Req	Cond. Beh. Req.	Mandatory
1.2 Diagnostic Trouble Code [yes, no] Hint: For safety relevant errors, a DTC shall be stored	Folder	-	Mandatory
[case yes] DTC Number = tbd	Information	-	Mandatory
[case yes] IF the signal is missing > XX s, THEN then a DTC shall be stored	Functional Req	Cond. Beh. Req.	Mandatory

Nicht testbar, Child muss mit Details verlinkt werden

Beispiel Template-Klasse

	Type	Classification	Protection
1 IN_Signal_Reqs Hint: In this template are input-signal relevant reqs are defined	Class	-	-
1.1 Error Reqs and Handling	Folder	-	Mandatory
1.1.1 Signal Range Check	Folder	-	Mandatory
IF the signal is out of range, THEN the value shall be estimated	Functional Req	Functional Goal	Proposal
1.1.2 Communication Timeout Test [yes / no] Hint: A communication timeout test shall be performed for all cyclic signals	Folder	-	Mandatory
[case yes] IF the signal is missing > XX ms, THEN the last signal shall be used	Functional Req	Cond. Beh. Req.	Mandatory
1.2 Diagnostic Trouble Code [yes, no] Hint: For safety relevant errors, a DTC shall be stored	Folder	-	Mandatory
[case yes] DTC Number = tbd	Information	-	Mandatory
[case yes] IF the signal is missing > XX s, THEN then a DTC shall be stored	Functional Req	Cond. Beh. Req.	Mandatory

Entscheidung muss im Child getroffen werden



Beispiel Template-Klasse

	Type	Classification	Protection
1 IN_Signal_Reqs Hint: In this template are input-signal relevant reqs are defined	Class	-	-
1.1 Error Reqs and Handling	Folder	-	Mandatory
1.1.1 Signal Range Check	Folder	-	Mandatory
IF the signal is out of range, THEN the value shall be estimated	Functional Req	Functional Goal	Proposal
1.1.2 Communication Timeout Test [yes / no] Hint: A communication timeout test shall be performed for all cyclic signals	Folder	-	Mandatory
[case yes] IF the signal is missing > XX ms, THEN the last signal shall be used	Functional Req	Cond. Beh. Req.	Mandatory
1.2 Diagnostic Trouble Code [yes, no] Hint: For safety relevant errors, a DTC shall be stored	Folder	-	Mandatory
[case yes] DTC Number = tbd	Information	-	Mandatory
[case yes] IF the signal is missing > XX s, THEN then a DTC shall be stored	Functional Req	Cond. Beh. Req.	Mandatory

Detail muss in Child definiert werden

Beispiel Abgeleitete Klasse

Parent-Klasse

	Type	Classification	Protection
3 CAN_IN_Signal_Reqs extends IN_Signal_Reqs Hint: In this class CAN input-signal relevant reqs are defined	Class	-	-
3.1 Error Reqs and Handling	Folder	-	Mandatory
3.1.1 Signal Range Check	Folder	-	Mandatory
IF the signal is out of range, THEN the value shall be estimated	Functional Req	Functional Goal	Proposal
3.1.2 Communication Timeout Test [yes / no] Hint: A communication timeout test shall be performed for all cyclic signals	Folder	-	Mandatory
[case yes] IF the signal is missing > XX ms, THEN the last signal shall be used	Functional Req	Cond. Beh. Req.	Mandatory
3.2 Diagnostic Trouble Code [yes] Hint: For any safety relevant error, a DTC shall be stored	Folder	-	Mandatory
DTC Number = tbd	Information	-	Mandatory
IF the signal is missing > **30 s**, THEN then a DTC shall be stored	Functional Req	Cond. Beh. Req.	Mandatory

VISION:EER

Beispiel Abgeleitete Klasse

	Type	Classification	Protection
3 CAN_IN_Signal_Reqs extends IN_Signal_Reqs Hint: In this class CAN input-signal relevant reqs are defined	Class	-	-
3.1 Error Reqs and Handling	Folder	-	Mandatory
3.1.1 Signal Range Check	Folder	-	Mandatory
IF the signal is out of range, THEN the value shall be estimated	Functional Req	Functional Goal	Proposal
3.1.2 Communication Timeout Test [yes / no] Hint: A communication timeout test shall be performed for all cyclic signals	Folder	-	Mandatory
[case yes] IF the signal is missing > XX ms, THEN the last signal shall be used	Functional Req	Cond. Beh. Req.	Mandatory
3.2 Diagnostic Trouble Code [yes] Hint: For any safety relevant error, a DTC shall be stored	Folder	-	Mandatory
DTC Number = tbd	ormation	-	Mandatory
IF the signal is missing > **30 s**, THEN then	nctional Req	Cond. Beh. Req.	Mandatory

Finalisierung der gemeinsamen Anforderungen für CAN-Signale

VISION:EER

Beispiel Klassen-Instanz

Signal Name = Instanz

	Type	Classification	Protection
4 Outdoor_Temp_signal - CAN_IN_Signal_Reqs Hint: In this class CAN input-signal relevant reqs are defined	Class	-	-
4.1 Error Reqs and Handling	Folder	-	Mandatory
4.1.1 Signal Range Check	Folder	-	Mandatory
IF the signal is out of range, THEN the value shall be estimated	Functional Req	Functional Goal	Proposal
4.1.2 Communication Timeout Test [yes] Hint: A communication timeout test shall be performed for all cyclic signals	Folder	-	Mandatory
IF the signal is missing > **100 ms**, THEN the last signal shall be used	Functional Req	Cond. Beh. Req.	Mandatory
4.2 Diagnostic Trouble Code [yes] Hint: For any safety relevant error, a DTC shall be stored	Folder	-	Mandatory
DTC Number = **0x122**	Information	-	Mandatory
IF the signal is missing > 30 s, THEN then a DTC shall be stored	Functional Req	Cond. Beh. Req.	Mandatory

VISION:EER

Beispiel Klassen-Instanz

	Type	Classification	Protection
4 Outdoor_Temp_signal - CAN_IN_Signal_Reqs Hint: In this class CAN input-signal relevant reqs are defined	Class	-	-
4.1 Error Reqs and Handling	Folder	-	Mandatory
4.1.1 Signal Range Check	Folder	-	Mandatory
IF the signal is out of range, THEN the value shall be estimated	Functional Req	Functional Goal	Proposal
4.1.2 Communication Timeout Test [yes] Hint: A communication timeout test shall be performed for all cyclic signals	Folder	-	Mandatory
IF the signal is missing > **100 ms**, THEN the last signal shall be used	Functional Req	Cond. Beh. Req.	Mandatory
4.2 Diagnostic Trouble Code [yes] Hint: For any safety relevant error, a DTC shall be stored	Folder	-	Mandatory
DTC Number = **0x122**	Information	-	Mandatory
IF the signal is missing > 30	Functional Req	Cond. Beh. Req.	Mandatory

Finalisierung der signalspezifischen Anforderungen

VISION:EER

Visioneer Tool Funktionen

Die 1. Version des Visioneer-Tools:
ist ein Add-On für Codebeamer

Tool Funktionen:

1. **Regeln für die Klassenvererbung**
2. **Prozess zur Erstellung abgeleiteter Klassen**
3. **Prozess für die Instanziierung von Klassen**
4. **Einhaltung der Klassenregeln und Error- Handling**

VISION:EER

1. Regeln für die Klassenvererbung

<u>Abgeleitete Klassen und Klasseninstanzen erben von ihren Eltern:</u>

- alle Items
- alle Fields (Attribute)
- alle Field-Values, außer

Tags, *Associations*, *Downstream References*, *Attachments/Comments*, *Spent Effort*, *Status values* werden nicht aus der Vorlage kopiert (von Codebeamer)

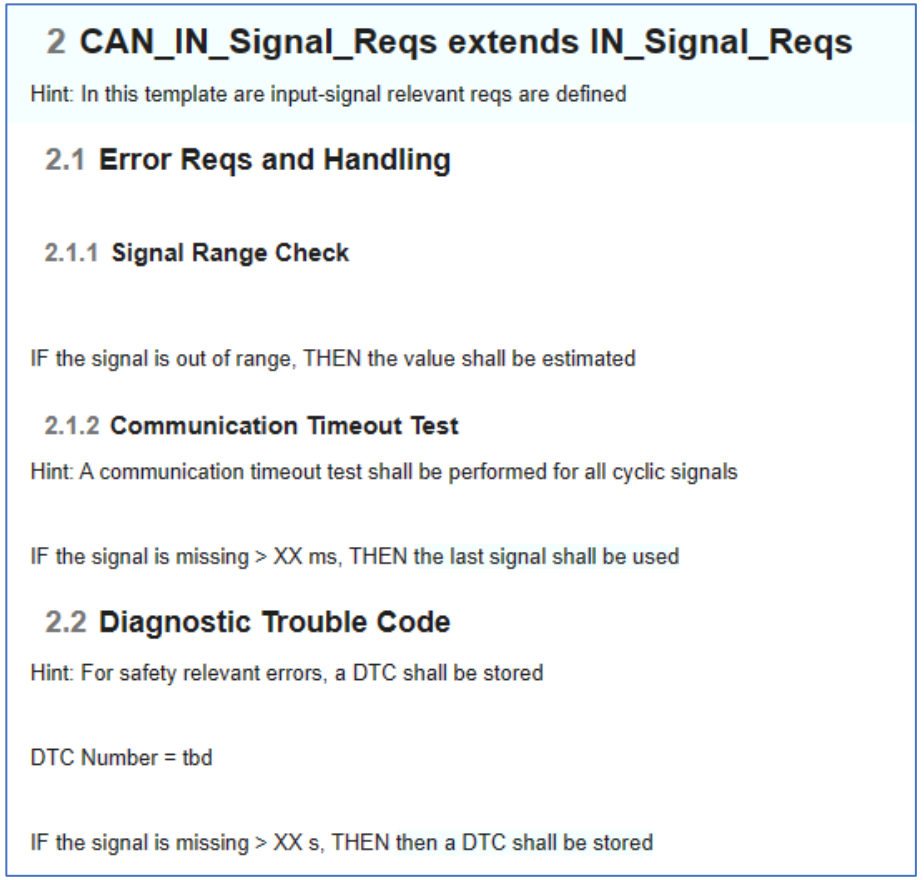
VISION:EER

2. Prozess für Abgeleitete Klassen

Abgeleitete Klassen können nur im Elementvorlagen-Tracker erstellt werden.

- Ein Ordner muss mit der folgenden Syntax erstellt werden:

- Nach dem Drücken der Synchronisationstaste



3. Prozess für Klasseninstanzen

- Um eine Klassen-Instanz zu erstellen, muss ein Ordner mit der folgenden Syntax erstellt werden:

▸ 4 Outdoor_Temp_signal - CAN_IN_Signal_Reqs

- Nach dem Drücken der Synchronisationstaste wird die Parent-Klasse gemäß den Vererbungsregeln erzeugt:

4 Outdoor_Temp_signal - CAN_IN_Signal_Reqs

Hint: In this class CAN input-signal relevant reqs are defined

4.1 Error Reqs and Handling

4.1.1 Signal Range Check

IF the signal is out of range, THEN the value shall be estimated

4.1.2 Communication Timeout Test [yes / no]

Hint: A communication timeout test shall be performed for all cyclic signals

4.2 Diagnostic Trouble Code [yes]

Hint: For any safety relevant error, a DTC shall be stored

DTC Number = tbd

IF the signal is missing > **30 s**, THEN then a DTC shall be stored

VISION:EER

4. Einhaltung der Klassenregeln und Error-Handling

Nach dem Drücken der Synchronisierung. klicken, jede abgeleitete Klasse und jede Klasseninstanz muss überprüft werden, ob es den Vererbungssregeln der Parents folgt:

- Ob ein **obligatorisches Element gelöscht** ist
- Ob Text, der nicht überschrieben werden darf, **überschrieben** ist
- Ob nur zulässige **case-relevante Reqs vererbt** sind
- Ob die Anzahl der **Multiplizitäten** im Bereich ihrer **Kardinalitäten** liegt
- Wenn alle **offenen Details** in Klasseninstanzen definiert sind

→ Wird ein Fehler festgestellt, dann wird er im Error-Field (Pflichtfeld) gespeichert

→ Wenn der Fehler korrigiert ist, wird das Error-Field wieder gelöscht

VISION:EER

Visioneer-Tool Versionen

<u>Version 1</u> (für Codebeamer) → verfügbar Mai 23

- *Reqs-Vorlagen, Klassenbehandlung und Überprüfung von Vererbungsregeln*

<u>Version 2</u> (for Codebeamer) → geplant am September 23

- *Durchführung aller fehlenden RIL-Tests und Testprotokollerstellung*

<u>Version 2</u> (for Doors) → geplant am Dezember 23

- *Gleiche Funktionen wie v2.0 für CB*

<u>Version 3</u> (for Doors and CB) → geplant Mär 24

- *Implementierung von MBSE-Schnittstellen*

VISION:EER

Vielen Dank für Ihre Aufmerksamkeit

www.visioneer.info