Visioneer Tool Handbook

for the Demo Version v1.0

6 September 2023

Contents:

1 How to get a Demo Version?

Important: The demo version v1.0 is an add-on for Codebeamer, which is installed on the Visioneer server

Hence, the demo version user does not need an own Codebeamer license,

but get access to a Codebeamer demo tracker, which contains examples and can be used as a playground (Spielwiese) for the implemented tool functions

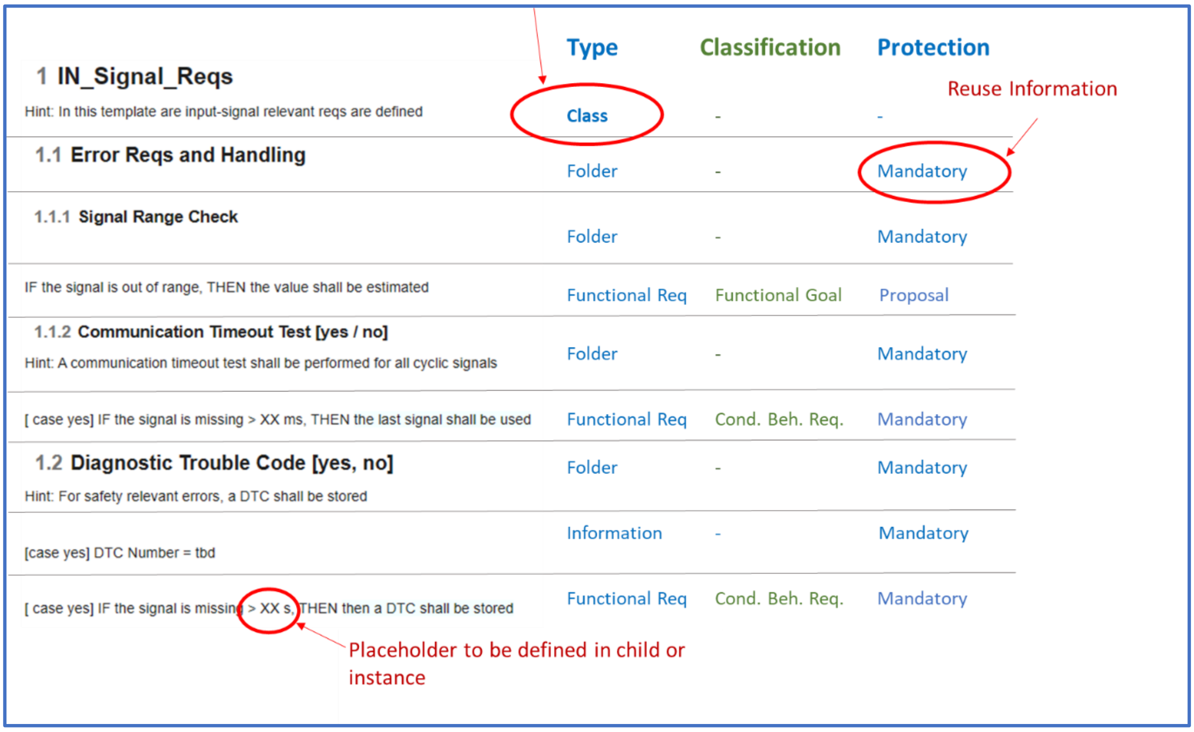
→. No download is required, we just need the E-mail address and then invite the requester as a demo version user

Simply send your demo user request by e-mail to: demo@visioneer.info

2 Implemented features

2.1 Creating Requirement Template Classes (Parent)

Description of all common complete/incomplete requirements and specification structures of a functional entity (*e.g. all signals*) in reusable form, as depicted in the following example:



	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

Reuse Information

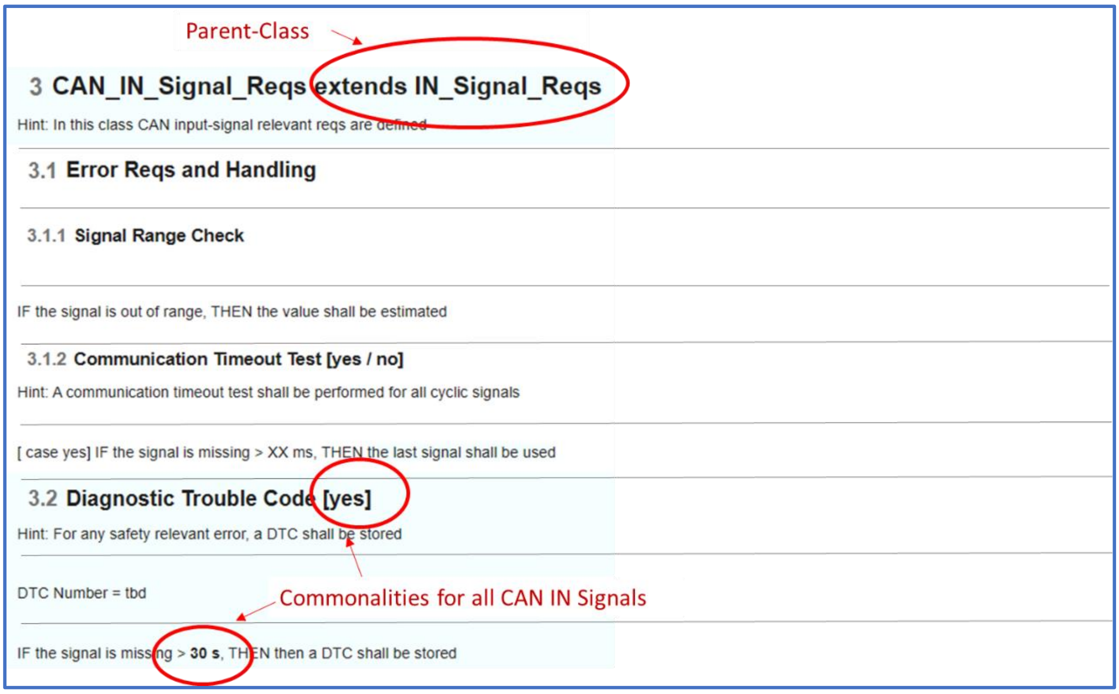
Placeholder to be defined in child or instance

2.2 Description of case-relevant requirements

Inheritance of the specific case-relevant requirements (depending on decision requirements)

2.3 Deriving Requirement Template Classes (Child)

Specialization of the **commonalities** of sub-entities (*e.g. all CAN signals*): Inheritance of the parent contents and completion, extension, overwriting or deletion of parts of the inherited items, as depicted in the following example:



Parent-Class

3 CAN_IN_Signal_Reqs extends IN_Signal_Reqs

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

3.1 Error Reqs and Handling

3.1.1 Signal Range Check

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

3.1.2 Communication Timeout Test [yes / no]

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

[case yes] IF the signal is missing > XX ms, THEN the last signal shall be used

3.2 Diagnostic Trouble Code [yes]

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

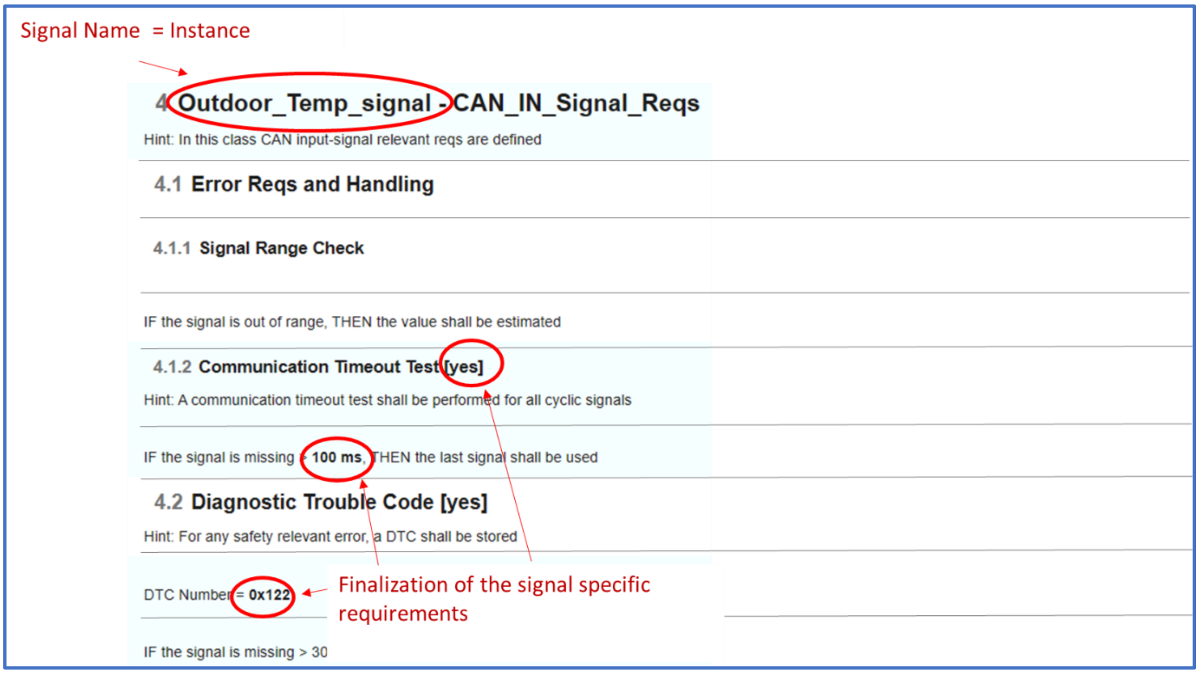
DTC Number = tbd

Commonalities for all CAN IN Signals

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

2.4 Instantiating Request Template Classes (Instance)

Reuse of the template classes for the specification creation, as depicted in the following example:



Signal Name = Instance

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]

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

IF the signal is missing > **100 ms**, THEN the last signal shall be used

4.2 Diagnostic Trouble Code [yes]

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

DTC Number = **0x122**

Finalization of the signal specific requirements

IF the signal is missing > 30

3 What Problems can be solved with this features

Generic requirements apply to (must be reused by) <u>each of the member</u> of a functional <u>entity</u>

--> **Reusable Requirements Libraries**

The following <u>Generic Requirements</u> exist in <u>any project</u> :

- Development Goals (e.g. an “economic system”)
- Standards or Regulations (e.g. Safety Regulations,..)
- Legal Requirements
- Security and IT requirements
- Component specific Requirements (e.g. for Embedded Systems, for ASICs,..)
- …

Vice versa, a template can be created for each entity of the system architecture

--> **Requirements System Kit**

According to the Single Source of Truth (SSOT), these may only be defined in one location. However, duplicates of the SSOT are unavoidable for describing the children and the individual entity members (instances)

➔ **<u>Automatic synchronization</u> must be performed**

It must contain all the necessary information regarding the inheritance rules (erasable, overwritten, etc.)

➔ **<u>Automatic verification</u> required, if inheritance rules are followed**

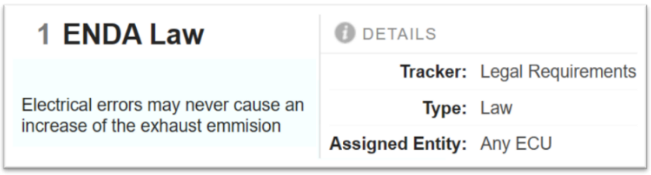
3.1 Example Application: Legal Requirments

From RE perspective **Legal Requirements** are:

- **Generic requirements**, that must be <u>assigned</u> to each of the <u>members</u> of an <u>entity</u> (e.g. for all signals)
- **Vague requirements**, that must be <u>linked</u> with derived <u>concrete requirements</u> (solutions)

<u>Example:</u> ENDA Law (simplified)

<u>Database for applicable laws:</u>



1 ENDA Law

Electrical errors may never cause an increase of the exhaust emmision

DETAILS

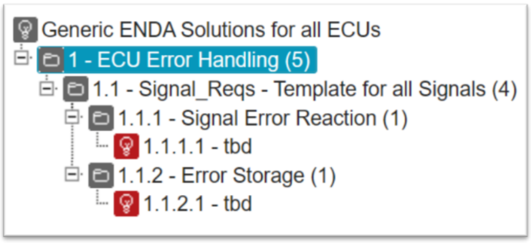
Tracker: Legal Requirements

Type: Law

Assigned Entity: Any ECU

Linked solutions

<u>Solutions-Library:</u>

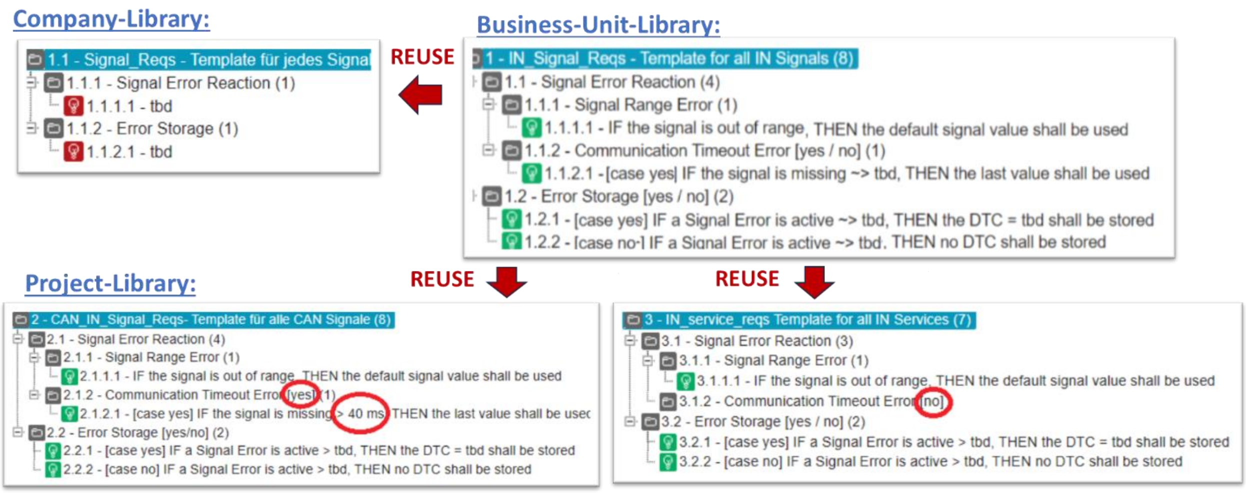


Generic ENDA Solutions for all ECUs

- 1 - ECU Error Handling (5)
  - 1.1 - Signal_Reqs - Template for all Signals (4)
    - 1.1.1 - Signal Error Reaction (1)
      - 1.1.1.1 - tbd
    - 1.1.2 - Error Storage (1)
      - 1.1.2.1 - tbd

In the example, there is a generic template for all signals, which secures that for each of the signals the Signal Error Reaction and Error Storage are defined, without defining the details how this shall be performed by using the placeholder *tbd.*

The generic template, which is in this example valid for the whole company, is then reused and specialized by various suborganizations:



Company-Library:

1.1 - Signal_Reqs - Template für jedes Signal
1.1.1 - Signal Error Reaction (1)
1.1.1.1 - tbd
1.1.2 - Error Storage (1)
1.1.2.1 - tbd

Business-Unit-Library:

1 - IN_Signal_Reqs - Template for all IN Signals (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 default 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 last 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

Project-Library:

2 - CAN_IN_Signal_Reqs- Template für alle CAN Signale (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 default signal value shall be used
2.1.2 - Communication Timeout Error [yes] (1)
2.1.2.1 - [case yes] IF the signal is missing > 40 ms, THEN the last value shall be usec
2.2 - Error Storage [yes/no] (2)
2.2.1 - [case yes] IF a Signal Error is active > tbd, 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

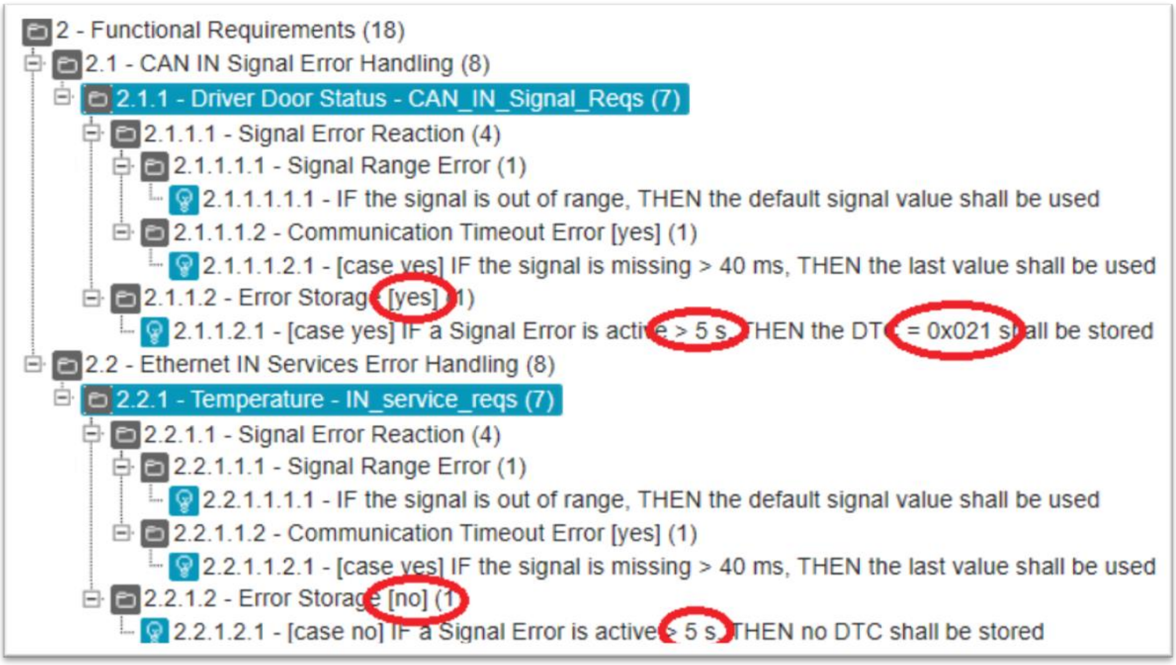
3 - IN_service_reqs Template for all IN Services (7)
3.1 - Signal Error Reaction (3)
3.1.1 - Signal Range Error (1)
3.1.1.1 - IF the signal is out of range, THEN the default signal value shall be used
3.1.2 - Communication Timeout Error [no]
3.2 - Error Storage [yes / no] (2)
3.2.1 - [case yes] IF a 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

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 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 the 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 its specific case decisions and values are fixed.

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

ECU Specification:



2 - Functional Requirements (18)
2.1 - CAN IN Signal Error Handling (8)
2.1.1 - Driver Door Status - CAN_IN_Signal_Reqs (7)
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 default 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 > 40 ms, THEN the last value shall be used
2.1.1.2 - Error Storage [yes] (1)
2.1.1.2.1 - [case yes] IF a Signal Error is active > 5 s, THEN the DTC = 0x021 shall be stored
2.2 - Ethernet IN Services Error Handling (8)
2.2.1 - Temperature - IN_service_reqs (7)
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 default signal value shall be used
2.2.1.1.2 - Communication Timeout Error [yes] (1)
2.2.1.1.2.1 - [case yes] IF the signal is missing > 40 ms, THEN the last value shall be used
2.2.1.2 - Error Storage [no] (1)
2.2.1.2.1 - [case no] IF a Signal Error is active > 5 s, THEN no DTC shall be stored

Note:

The shown reuse process is required for any Generic Requirement in any project.

But:

A controlled reuse of requirement templates can only can be achieved by inheritance with object-oriented methods (as the controlled reuse of SW code)

Alternatively, the whole process must be performed **manually**, which means enormous efforts for (manually) **editing**, **linking** and **reviewing** each of the reused items.

Additionally: There is still risk, that **reuse** is incorrect (e.g. because the reviews are not repeated obligatory in each Requirement Freeze)

➔ **Tool supported reuse required**

4 Startup and configuration

After registering as demo user, you will receive a mail that contains your:

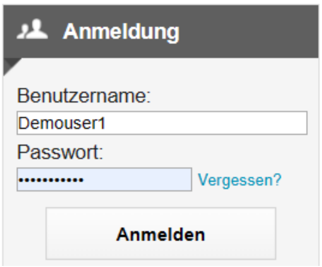
Codebeamer User Name = *Demouser1*

Password = *Password123*

Additionally, a link to our demo version implemented on Codebeamer on our server:

Intland codebeamer

After clicking the demo user can login:

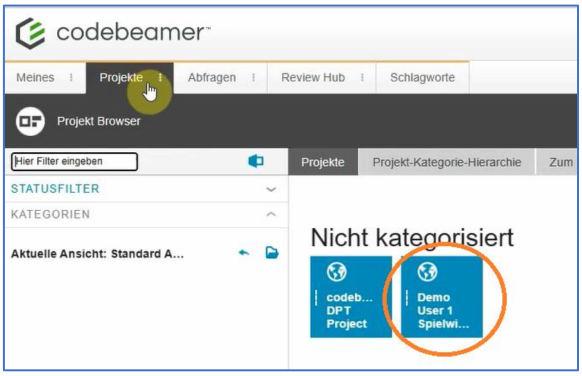


Anmeldung

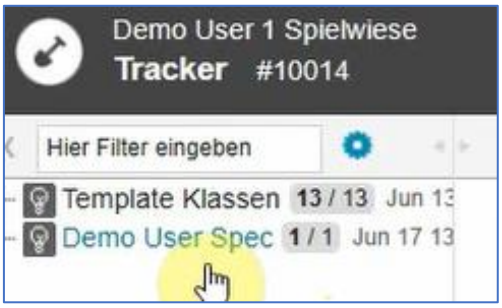
Benutzername: Demouser1

Passwort: ••••••••••• Vergessen?

After login, the user project (Spielwiese) shall be opened:



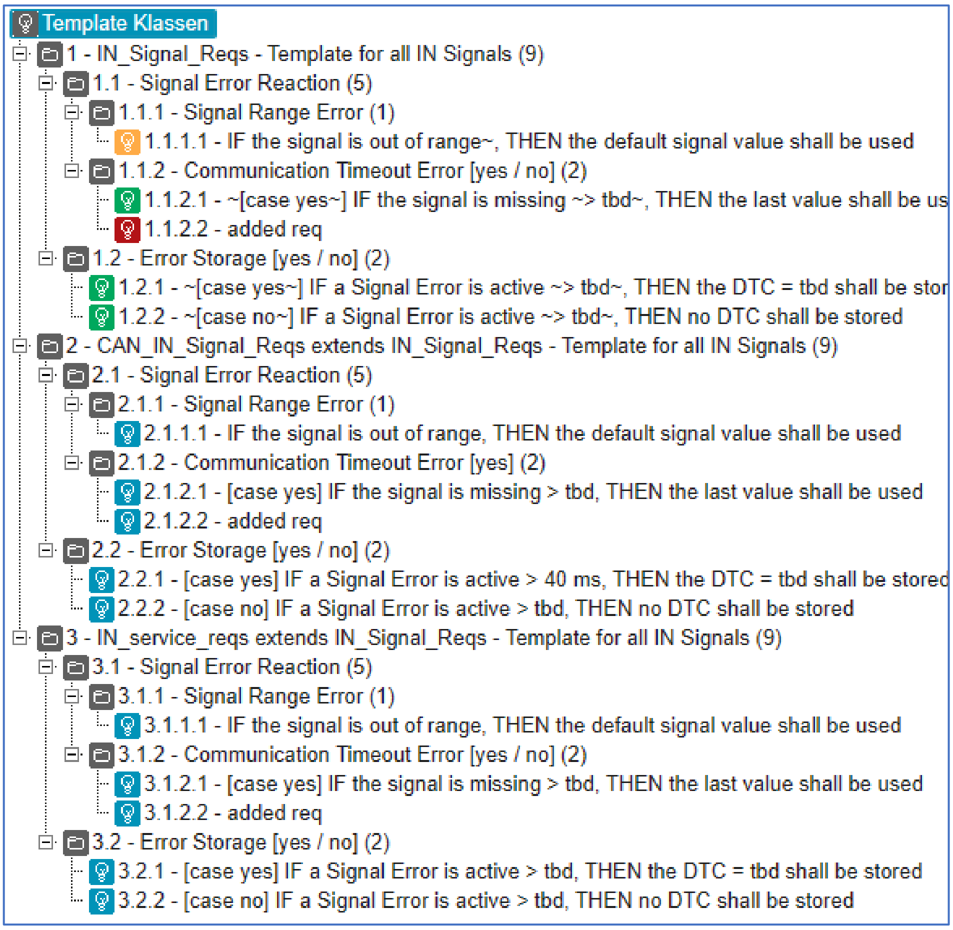
In this project the following tracker (= specification) are preinstalled:



Template Klassen 13 / 13 Jun 13

Demo User Spec 1 / 1 Jun 17 13

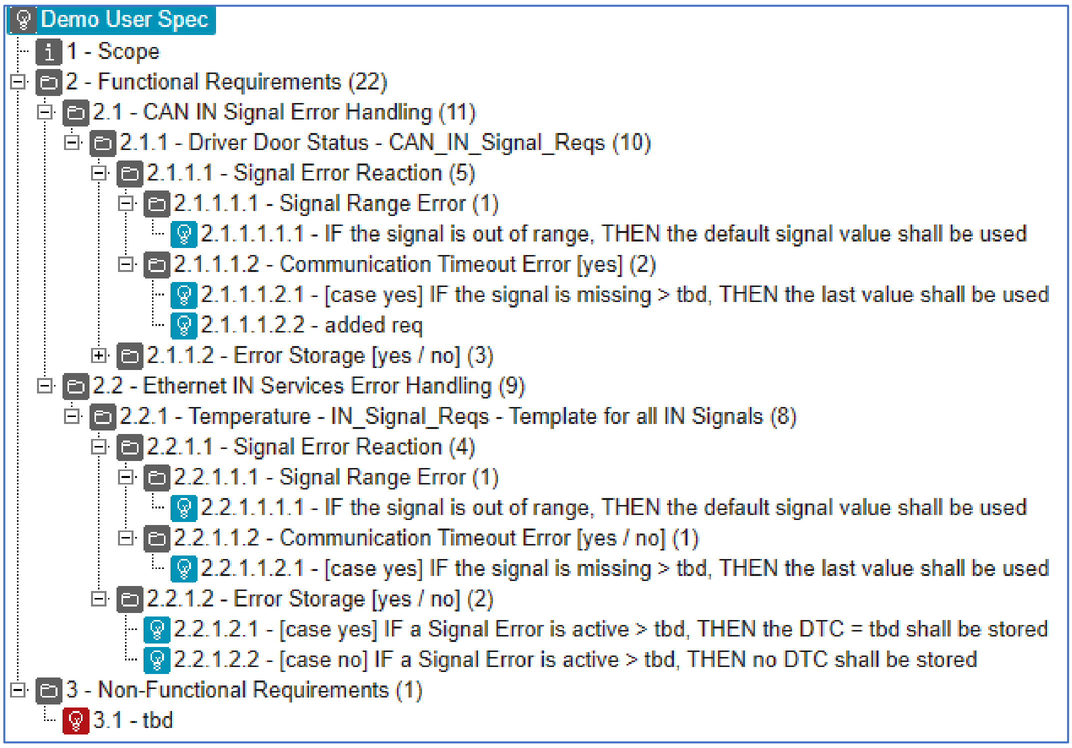
The tracker “Template Klassen” contains examples of parent and derived template classes:



Template Klassen

- 1 - IN_Signal_Reqs - Template for all IN Signals (9)
  - 1.1 - Signal Error Reaction (5)
    - 1.1.1 - Signal Range Error (1)
      - 1.1.1.1 - IF the signal is out of range~, THEN the default signal value shall be used
    - 1.1.2 - Communication Timeout Error [yes / no] (2)
      - 1.1.2.1 - ~[case yes~] IF the signal is missing ~> tbd~, THEN the last value shall be us
      - 1.1.2.2 - added req
  - 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 stor
    - 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 - Template for all IN Signals (9)
  - 2.1 - Signal Error Reaction (5)
    - 2.1.1 - Signal Range Error (1)
      - 2.1.1.1 - IF the signal is out of range, THEN the default signal value shall be used
    - 2.1.2 - Communication Timeout Error [yes] (2)
      - 2.1.2.1 - [case yes] IF the signal is missing > tbd, THEN the last value shall be used
      - 2.1.2.2 - added req
  - 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 - Template for all IN Signals (9)
  - 3.1 - Signal Error Reaction (5)
    - 3.1.1 - Signal Range Error (1)
      - 3.1.1.1 - IF the signal is out of range, THEN the default signal value shall be used
    - 3.1.2 - Communication Timeout Error [yes / no] (2)
      - 3.1.2.1 - [case yes] IF the signal is missing > tbd, THEN the last value shall be used
      - 3.1.2.2 - added req
  - 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

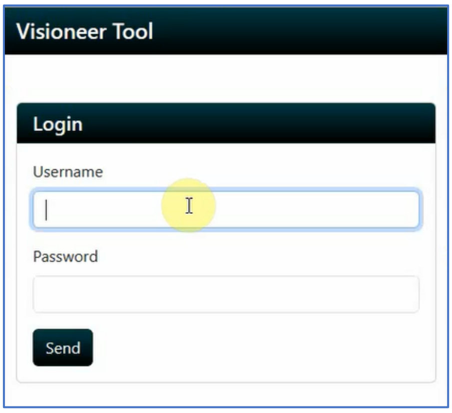
The tracker “Demo User Spec” contains examples for a controlled reuse of template classes:



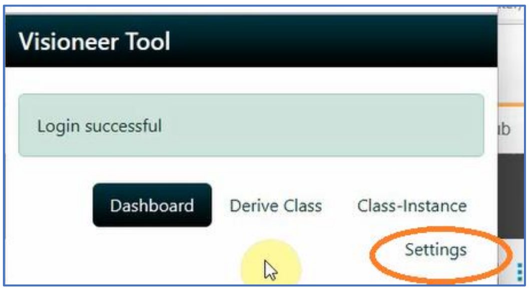
Demo User Spec

- 1 - Scope
- 2 - Functional Requirements (22)
  - 2.1 - CAN IN Signal Error Handling (11)
    - 2.1.1 - Driver Door Status - CAN_IN_Signal_Reqs (10)
      - 2.1.1.1 - Signal Error Reaction (5)
        - 2.1.1.1.1 - Signal Range Error (1)
          - 2.1.1.1.1.1 - IF the signal is out of range, THEN the default signal value shall be used
        - 2.1.1.1.2 - Communication Timeout Error [yes] (2)
          - 2.1.1.1.2.1 - [case yes] IF the signal is missing > tbd, THEN the last value shall be used
          - 2.1.1.1.2.2 - added req
      - 2.1.1.2 - Error Storage [yes / no] (3)
  - 2.2 - Ethernet IN Services Error Handling (9)
    - 2.2.1 - Temperature - IN_Signal_Reqs - Template for all IN Signals (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 default 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 last 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

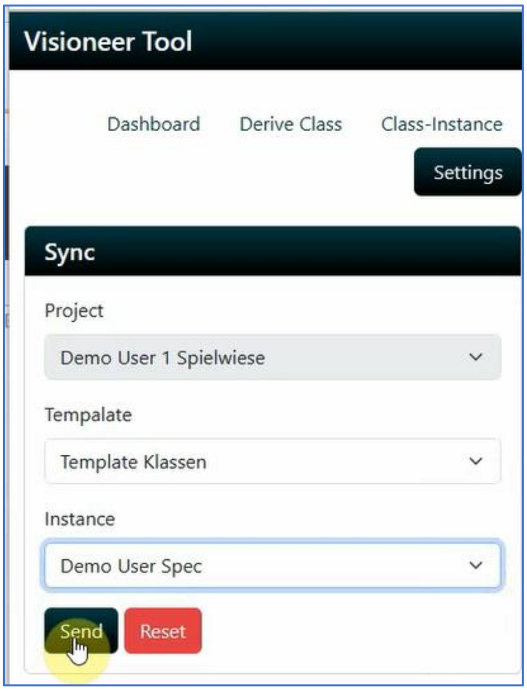
Then the user must login into the Visioneer AddOn (with the CB user data):



After login, the Settings button must ne pressed:


Login successful

To configure the tool, the following data must be selected and then the Send button must be pressed:



Project: Demo User 1 Spielwiese

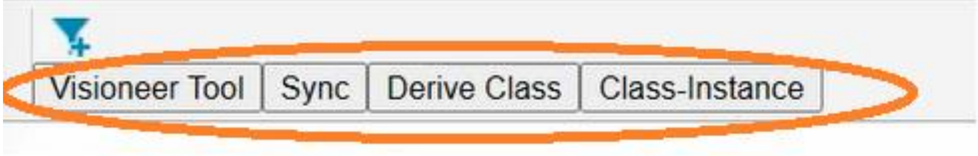
Tempalate: Template Klassen

Instance: Demo User Spec

Note: In later versions of the Visioneer Tool, more template tracker (e.g. from different organisation level) can be selected. Also more specification can be selected for instanciations.

5 Tool Operations

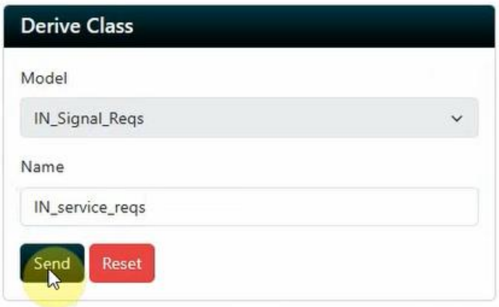
The tool-functions can be executed by pressing one of the following buttons:



Button	Function
Visioneer Tool	Tool Configuration (see previous chapter)
Sync	Synchronization of all children and instances (if the parent contents has changed)
Derive Class	A derived class is created (only in template tracker)
Class-Instance	A class-instance is created (only in specification tracker)

5.1 Creating a derived class

After pressing the derive class button, in the template tracker the **parent class** (Model) must be selected, the **name** of the new derived class must be defined and the Send button must be pressed

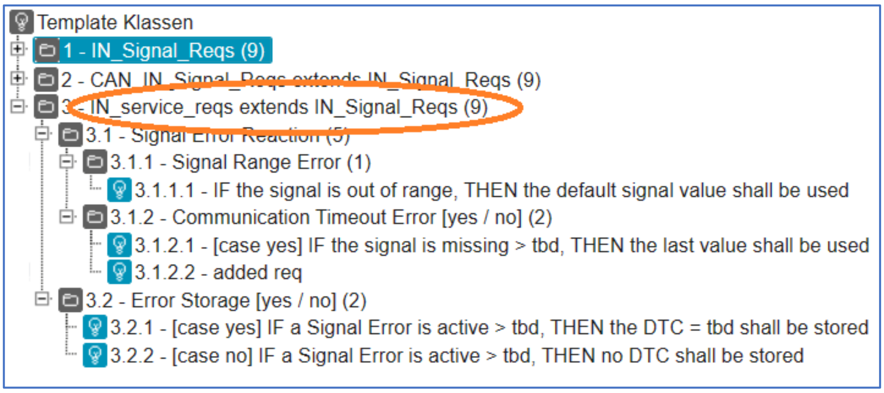


Derive Class

Model: IN_Signal_Reqs

Name: IN_service_reqs

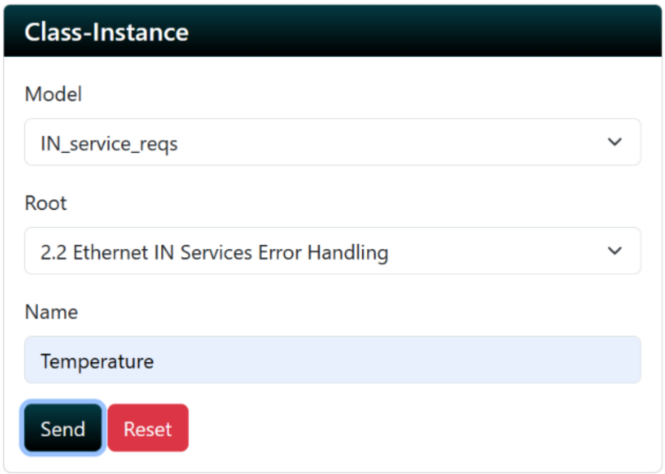
Then the new derived class is created:



```
Template Klassen
  1 - IN_Signal_Reqs (9)
  2 - CAN_IN_Signal_Reqs extends IN_Signal_Reqs (9)
  3 - IN_service_reqs extends IN_Signal_Reqs (9)
    3.1 - Signal Error Reaction (5)
      3.1.1 - Signal Range Error (1)
        3.1.1.1 - IF the signal is out of range, THEN the default signal value shall be used
      3.1.2 - Communication Timeout Error [yes / no] (2)
        3.1.2.1 - [case yes] IF the signal is missing > tbd, THEN the last value shall be used
        3.1.2.2 - added req
    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
```

5.2 Creating a class-instance

After pressing the class-instance button, in the specification tracker the **parent class** (Model) must be selected, the the **root-directory** must be selected for new chapter, the **name** of the new instance must be defined and the Send button must be pressed.



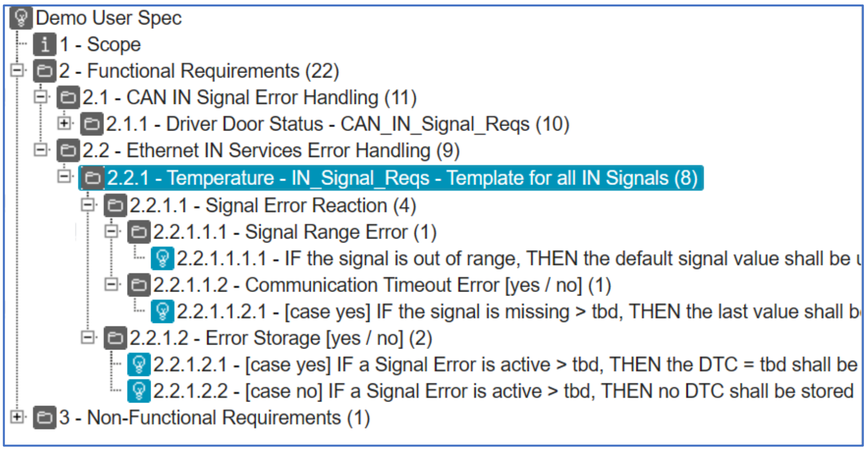
Class-Instance

Model: IN_service_reqs

Root: 2.2 Ethernet IN Services Error Handling

Name: Temperature

Then the new class-instance is created as subchapters in the specification:



Demo User Spec
1 - Scope
2 - Functional Requirements (22)
2.1 - CAN IN Signal Error Handling (11)
2.1.1 - Driver Door Status - CAN_IN_Signal_Reqs (10)
2.2 - Ethernet IN Services Error Handling (9)
2.2.1 - Temperature - IN_Signal_Reqs - Template for all IN Signals (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 default signal value shall be u
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 last value shall b
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
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)

6 Changes in the Parent-Class

The following changes in the parent classes are automatically inherited by the children and instances:

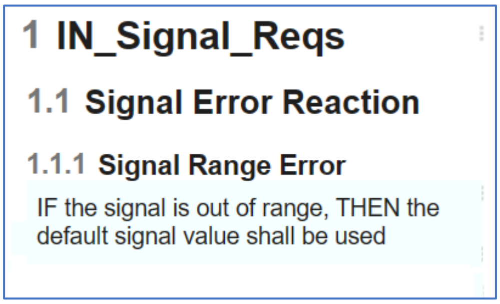
- Adding requirements
- Deletion of requirements
- Modification of requirement structures
- Modification of the textual description
- Modification of attribute-field content or links

Note: If the textual description of a parent item is changed, then the previous text in the children and instances is not deleted:

- The item is overwritten with the changed parent text → Status Auto Generated
- Additionally the old requirement is available in an new item → Status rejected

Example:

The following change is performed in the parent class:



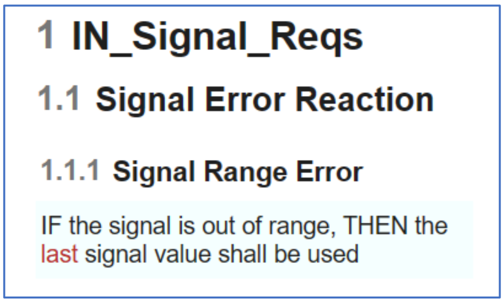
1 IN_Signal_Reqs

1.1 Signal Error Reaction

1.1.1 Signal Range Error

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

→



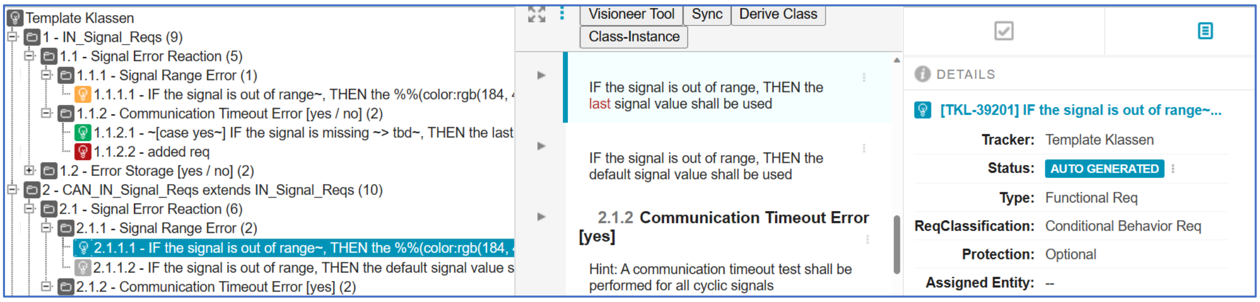
1 IN_Signal_Reqs

1.1 Signal Error Reaction

1.1.1 Signal Range Error

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

Then after pressing the Sync button the new item with status Auto Generated (blue) is created in the children and the old item is still existing, but has the status Rejected (grey):



Template Klassen
1 - IN_Signal_Reqs (9)
1.1 - Signal Error Reaction (5)
1.1.1 - Signal Range Error (1)
1.1.1.1 - IF the signal is out of range~, THEN the %%(color:rgb(184,
1.1.2 - Communication Timeout Error [yes / no] (2)
1.1.2.1 - ~[case yes~] IF the signal is missing ~> tbd~, THEN the last
1.1.2.2 - added req
1.2 - Error Storage [yes / no] (2)
2 - CAN_IN_Signal_Reqs extends IN_Signal_Reqs (10)
2.1 - Signal Error Reaction (6)
2.1.1 - Signal Range Error (2)
2.1.1.1 - IF the signal is out of range~, THEN the %%(color:rgb(184,
2.1.1.2 - IF the signal is out of range, THEN the default signal value s
2.1.2 - Communication Timeout Error [yes] (2)

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

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

2.1.2 Communication Timeout Error [yes]

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

[TKL-39201] IF the signal is out of range~...

Tracker: Template Klassen
Status: AUTO GENERATED
Type: Functional Req
ReqClassification: Conditional Behavior Req
Protection: Optional
Assigned Entity: --

More information about the demo version is available in our videos on our homepage:

www.visioneer.info