# SOX Version 3.1

# Documentation

ENCO

## Table of Content

# 1. Important changes - What is new?

This document provides an overview on the most important novel features and necessary information about the new license mechanism.

<u>**New features of SOX v3.1:**</u>

- **New SOX Module Test (TM):** The completely new module Test supports specifying and managing Test Specifications and Test Runs and their linkage to requirements. SOX enables full traceability among the whole engineering process, from requirements to tests.

- **Revised Module Hazard and Risk Analysis (HARA):** The module HARA has been completely reworked with improved structure, presentation, and usability. The HARA module now delivers additional efficiency gains by supporting reuse of events (e.g. effects, hazards) via catalogs. Included permutation algorithms improve the general handling of rated scenarios and additionally provide automatic restrictions to provide a manageable overview of all rated scenarios. Adaptable standard catalogs for the automotive driving scenarios are now delivered as default as well.

- **Version management:** Users can move back in project history and view any previous version of a project. Previous version can be fully loaded into the workspace to inspect all details using all available SOX views and editors. Versions can be created manually on requested specific times.

- **Improved Module System Designer:** Integration of the latest version of Papyrus UML/ SysML tool (Papyrus Oxygen 3.1), providing various usability and stability improvements.

- **Enhanced Synchronization across Documents:** Enhanced analysis of differences between FMEA documents and System Design as well as between FMEA and FMEDA documents. Automated adaptation of detected differences on demand.

- **Custom Process Support for Collaboration / State machine:** Enhanced and customizable support for setting the status of work products, such as requirements, component definitions or hazard definitions. Easy customization of states and permitted state transitions using state machine graphs. Specific states of elements can now also be assigned to assigned roles (SOX Users).**Traceability:** As all document content is unique over the whole project, any element (e.g., a requirement) can be traced over the whole project, e.g., finding all elements related to a requirement.

- **Trace Graph:** Trace graph providing a visual representation of the SOX Trace View, allowing users visual inspection of all types of links between elements within the same or different documents.

- **Multiple Repositories per Server:** Possibility to run multiple repositories on the same SOX server using different ports.

- **Note:** Server must be updated due to migration procedures. The instructions are delivered in a separate documentation for MySQL Servers available in the EnCo Download management center.

  https://www.enco-software.com/en/support/downloads/

- **Performance and Stability:** Significant performance improvements for server-based repositories. Significant performance improvements for FMEA document editing. Advanced data integrity checks via checksum calculations for project import/export and catalogues.

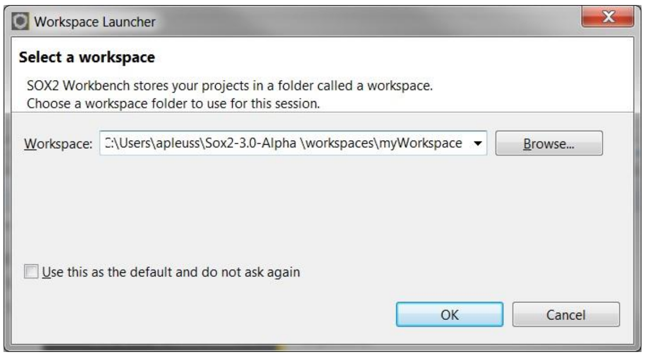
- **Various Minor Improvements:** Every existing SOX module has been under steady development. Smaller and greater application improvements are made to improve the general efficiency of SOX. For instance, copy requirements across different modules or manage the order of failure mode splits.

- **New Welcome Screen:** SOX starts with a Welcome Screen that allows to jump directly to the most important starting and administration tasks, tutorials, user help and information.

- **SSL Encryption:** Privacy protection assured between client and license server, respectively client and download server.

## 2. How to install and start SOX

- Download the SOX v3.1 set up file (exe-File).
- Install SOX v3.1 by opening SOX v3.1 set up file
- Use the application file to start SOX ("SOX.exe" in Windows).
- The language of SOX (English or German) depends on the default operating system language. (The language can be changed by starting SOX with a command line parameter *–nl en* (for English) or *–nl de* (for German). On Windows operating system, a command line option can be specified by creating a shortcut to *SOX.exe* in Windows Explorer, open its properties, and add the parameter at the end of the *target* property.)
- When SOX starts, you need to select a workspace. Create a new workspace by providing a **new** folder location on your computer where SOX can store data.


Workspace Launcher

Select a workspace

SOX2 Workbench stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

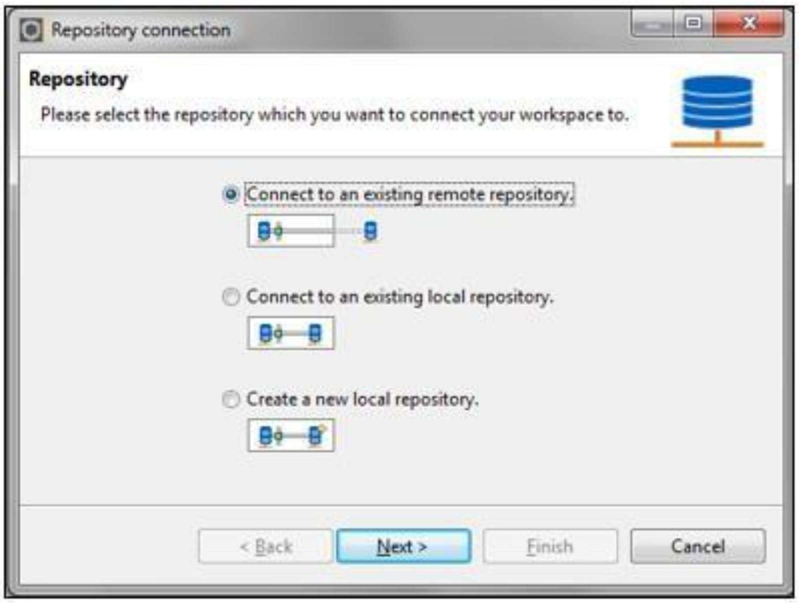
Workspace: C:\Users\apleuss\Sox2-3.0-Alpha \workspaces\myWorkspace

**Workspace:** *Contains internal data about connection to a repository and the projects that you have imported from this repository. Each workspace is associated with one repository. You can create multiple workspaces on your computer corresponding to multiple repository connections. Restart SOX to switch between workspaces or to create a new workspace.*

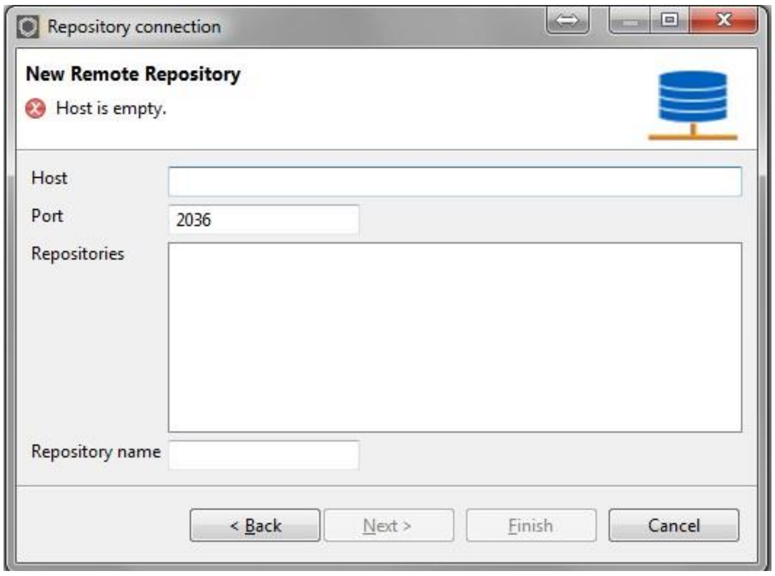
**Repository:** *Contains the actual projects. If you connect to an existing repository, you can then import its contained projects.*

- ***Local repository:*** *A repository stored locally on your computer. This option is easy for testing SOX but it does not allow you to share or access data or catalogues from/with users on other computers (except via import/export).*
- ***Remote repository:*** *A repository on a central server for parallel multi-user access and data sharing. This requires installation of a SOX server, e.g., by your system administrator.*

- After creating a new workspace you need to select the repository connection associated with that workspace. For first testing, create a new local repository (see third option below).
  Options:


Repository connection
Repository
Please select the repository which you want to connect your workspace to.
Connect to an existing remote repository.
Connect to an existing local repository.
Create a new local repository.
< Back
Next >
Finish
Cancel

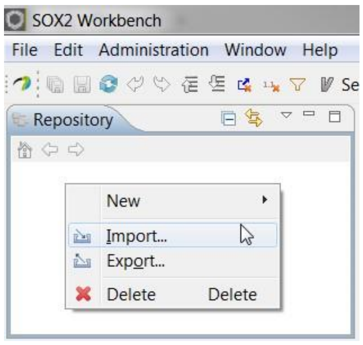
  - Connect to an existing remote repository: Ask your system administrator for the host name and the port of the SOX server in your organization, if available. The standard port is 2036. Then select one of the repositories on the specified server.



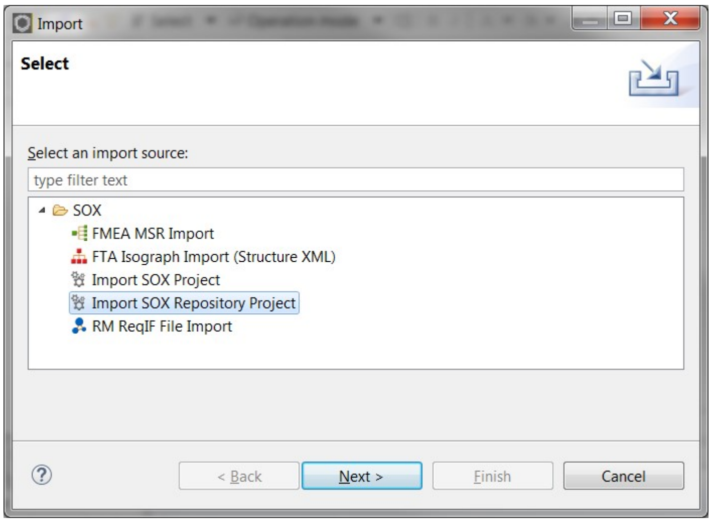
- ○ Connect to an existing local repository: Point to a folder on your computer to connect to an existing local repository and select from those found at this location.
- ○ Create a new local repository: Create/select a folder on your computer where SOX can store data and specify any repository name of your choice.

- If you connect to an existing repository, your workspace is initially empty. You can import existing projects from that repository by taking the following steps:

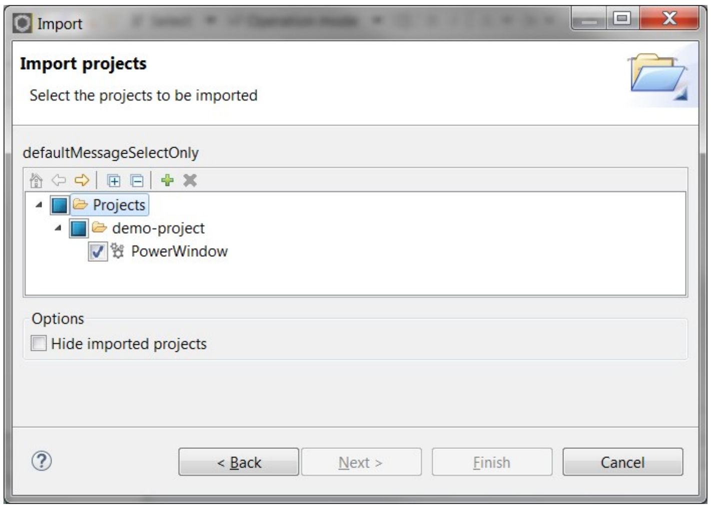
- ○ Right-click on repository window and select 'Import'.



- ○ Select 'Import SOX Repository Project'.



- Select the project to import, if available. (Expand the folder structure to see projects in subfolders)



- The instructions on how to install a SOX server to host remote repositories are provided in a separate document in the delivery package.

## 3. License Management

With the release of SOX v.3.1. a license server has been installed which allows a new and easier license management for the customer.

There are different license types and possibilities to work with SOX according to your internet connectivity.

1. Online
   - **1a) Floating license**
   - **1b) Node license**
2. Offline
   - **2a) Check out license**
   - **2b) Commuted license**

3 **Internal License Server**

### 1a) Floating license

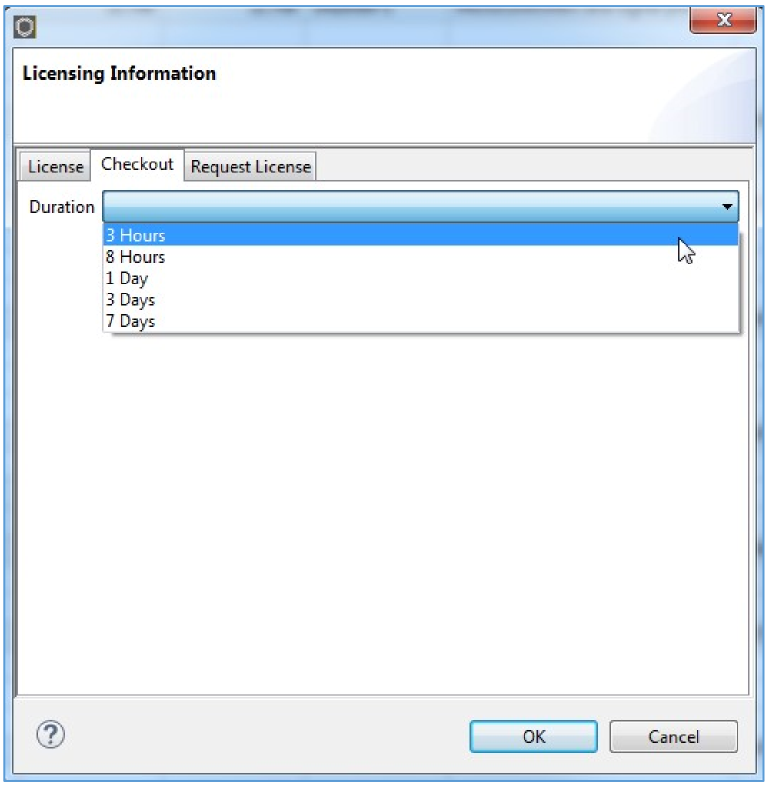
- Every user activates the same floating key and can then login into the system if the defined purchased number of floating licenses is not exceeded at this moment.
- The SOX-user must not give the key away to third parties as the key allows everyone to use this float license.
- If SOX is not running properly anymore (e.g., because the user has just closed his/her notebook, has no internet connection), the user is automatically logged-out on the server after 5 minutes.

## 1b) Node license

- Each key is bound to the hardware ID on the computer where the key is activated.
- It can be activated multiple times on the same hardware (e.g., if a user has deleted his/her SOX installation for some reason he/she can just activate it again) but not on a different hardware.
- User should not unintentionally activate multiple keys on the same machine as this would bind all these keys to the same hardware ID.

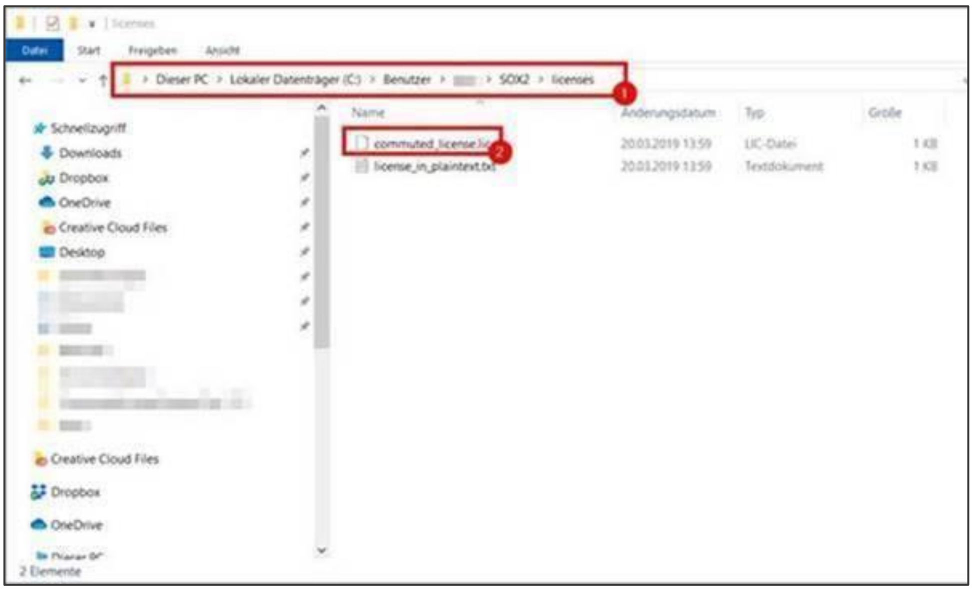
## 2a) Check out license

- User can check-out a license for a certain period of time (e.g. 1 day or 3 days) and afterwards work offline for this period of time.
- **Users should not forget that they must check-out the license before they are offline.**
- In case of a floating license, the user is "logged-in" for the period of time, i.e., checking out a license reduces the available license number for the company for this time of its duration. In this case please pay attention to return the checkout. Returning the checkout license is not possible when the client if offline.



**2b) Commuted license**

- Each key is bound to the hardware ID on the computer where the key is activated. Please provide us with your hardware ID while requesting this license type. Your hardware ID can be found at the license tab at the SOX license manger by clicking the copy ID to clipboard button.
- After this license type was requested, you will receive a license key file vial e-mail. The license key file must be placed locally at the user directory.
  (C:) > User > Username > SOX2 > licenses



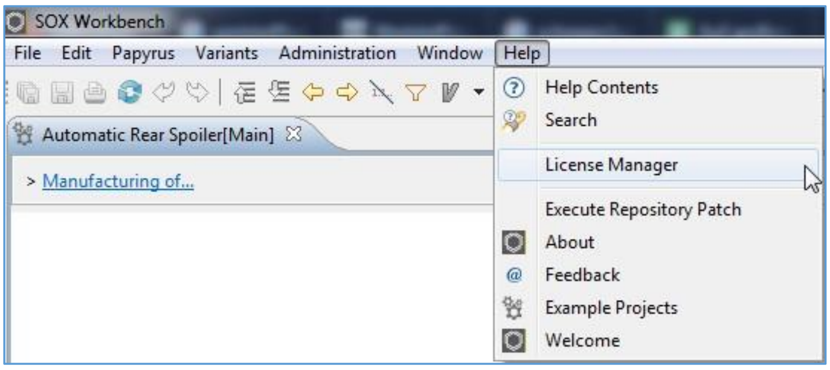
**3 Internal License Server**

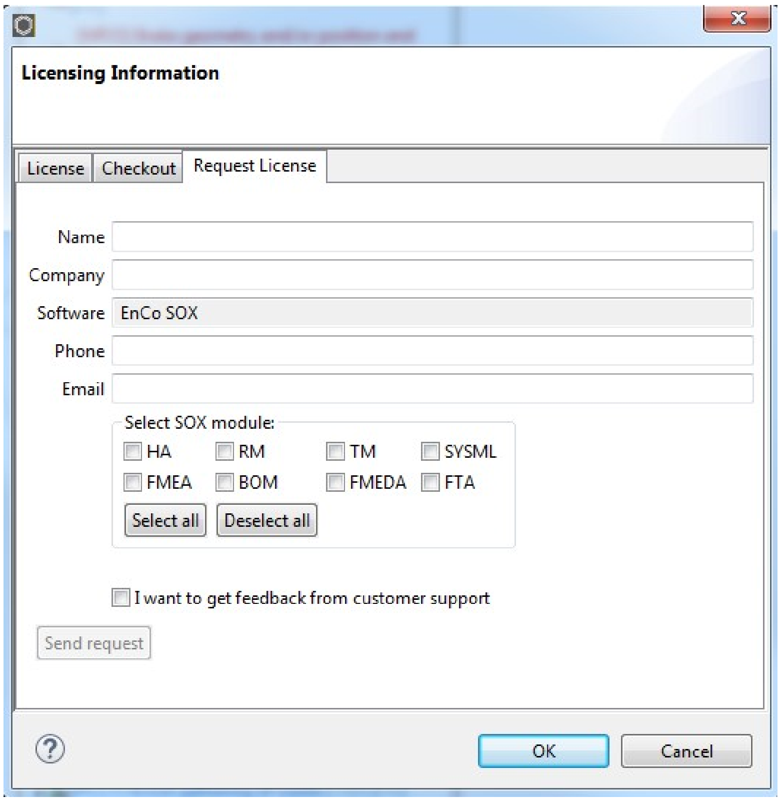
With the internal SOX license server you can organize your licenses by yourself. For this you need to install the license server on your systems. The license server installation files can be found at the download section on the EnCo Service Desk: https://www.enco-software.com/en/support/downloads/

You will also receive a key that activates the number of the licenses which you have paid for. Furthermore, you have several reporting possibilities to see which licenses are in use.

## License request

- If no valid license key is available, please request a new license within SOX using help – license manager.




**Licensing Information**

License | Checkout | Request License

| | |
|---|---|
| Name | |
| Company | |
| Software | EnCo SOX |
| Phone | |
| Email | |

Select SOX module:

| | | | |
|---|---|---|---|
| ☐ HA | ☐ RM | ☐ TM | ☐ SYSML |
| ☐ FMEA | ☐ BOM | ☐ FMEDA | ☐ FTA |

Select all · Deselect all

☐ I want to get feedback from customer support

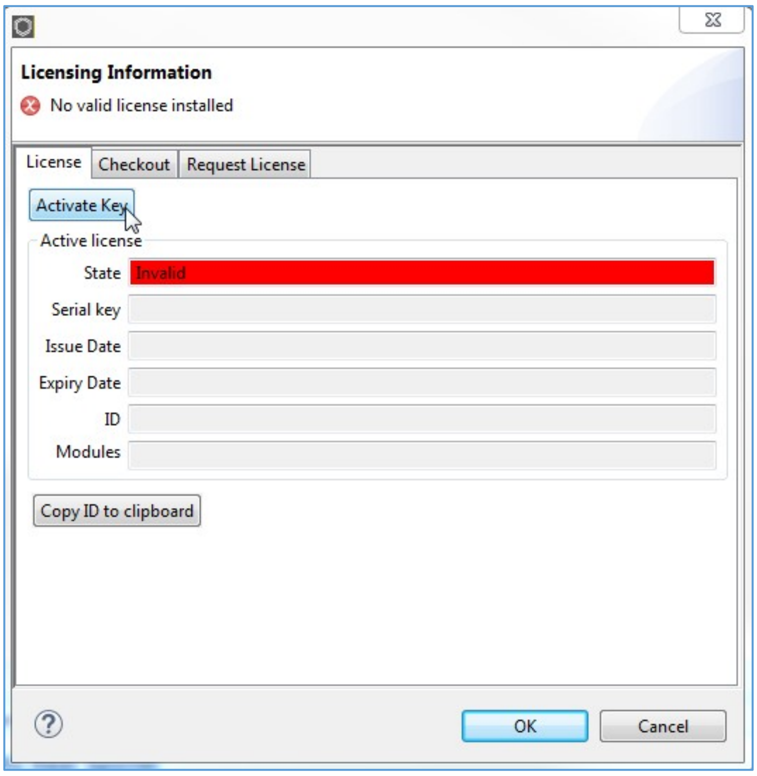
Send request

OK · Cancel

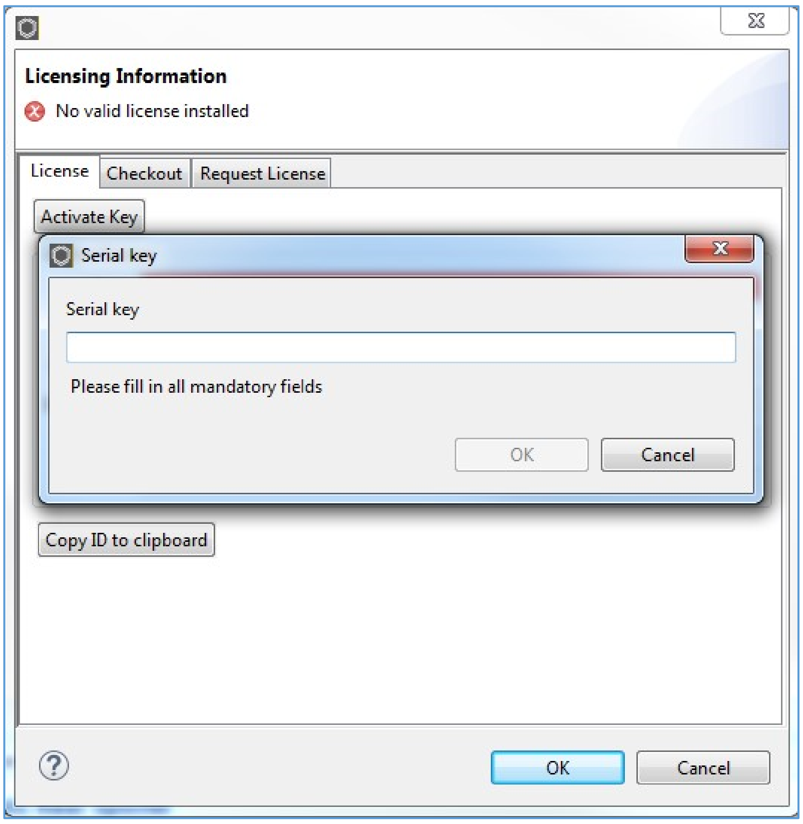
- Please fill out mandatory inputs such as name, company, e-mail and requested SOX modules.
- Select send request.

## Activate license key

- Once a license is requested EnCo will send you a valid license key which needs to be activated.
- Please insert your license key by selecting Activate Key.



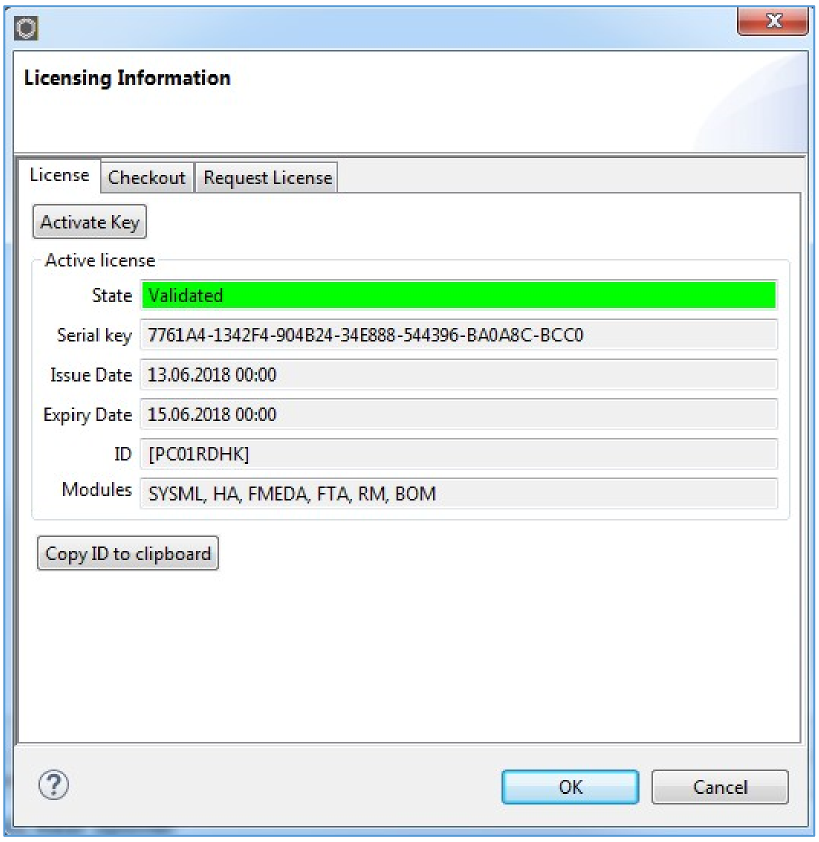
State: Invalid
Serial key:
Issue Date:
Expiry Date:
ID:
Modules:



Serial key

Please fill in all mandatory fields

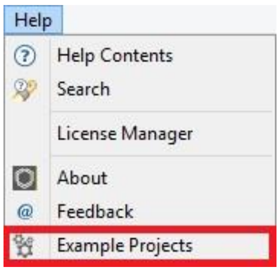
- License is now activated.
- As license information serial key, issue date, expiry date, client computer ID (MAC) and unlocked SOX modules are shown.



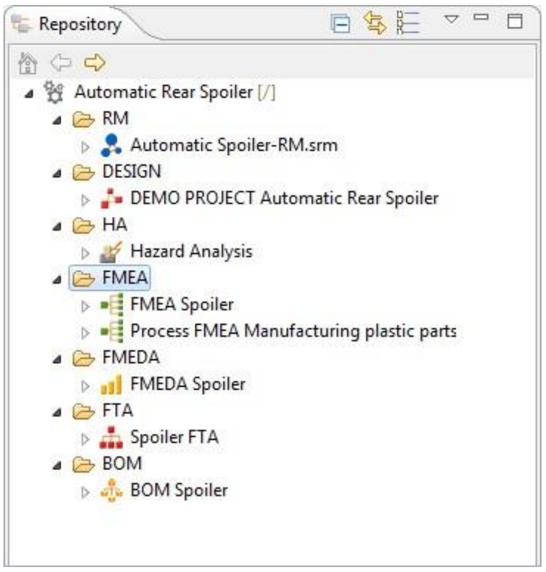
| | |
|---|---|
| State | Validated |
| Serial key | 7761A4-1342F4-904B24-34E888-544396-BA0A8C-BCC0 |
| Issue Date | 13.06.2018 00:00 |
| Expiry Date | 15.06.2018 00:00 |
| ID | [PC01RDHK] |
| Modules | SYSML, HA, FMEDA, FTA, RM, BOM |

## 4. Demo Project – Import and Description

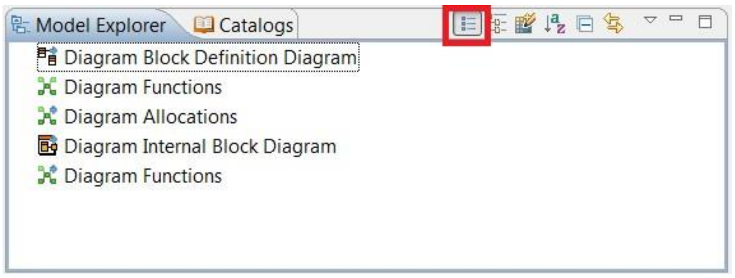
- In the menu bar, click on ´Help´, then ´Example Projects´.



- Select 'Automatic Rear Spoiler'. Click on ´Next´.
- As target folder, you can use the predefined 'projects' folder or create a custom folder. Click on ´Next´.

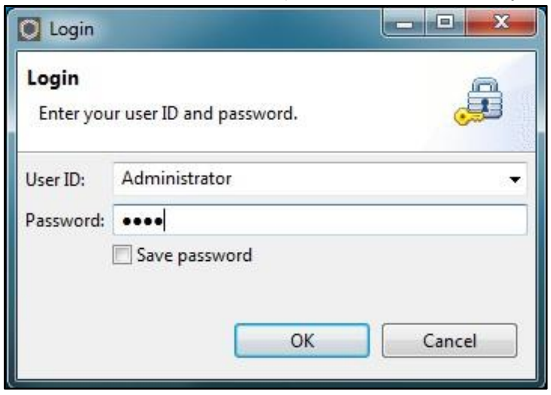


- Once the DEMO project is loaded, all modules appear.
- The DEMO project ‘Automatic Rear Spoiler´ contains sample documents for Requirements, System Designer, Hazard and Risk Analysis, FMEA, Reliability (BOM), FMEDA, and FTA.
- To open a document, double-click on the document in the “Repository” view.
- The SysML diagrams can be found in the “Model Explorer” view in the package ´diagrams´ within the SysML 1.4 model. Double-click on a diagram to open it. You can filter the “Model Explorer” view to show all available diagrams only using the “Show diagrams” toggle button in the toolbar of the “Model Explorer” view.

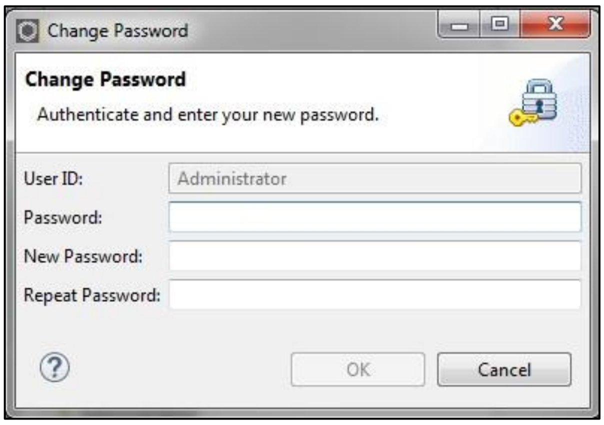


## 5. User Administration

- Type in a User ID (*Administrator* or *SoxUser*) and confirm with password: *0000*.



- In SOX there are two user roles available: *Administrator* and *User*.
- The Administrator is able to add new users and delegate tasks.
- The function "Change password" allows the Administrator and Users to create and change individual passwords.



Change Password

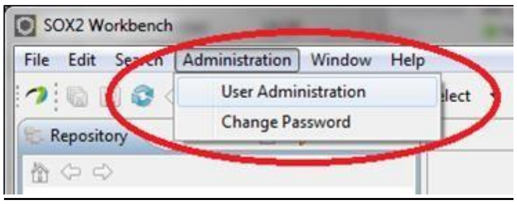
**Change Password**

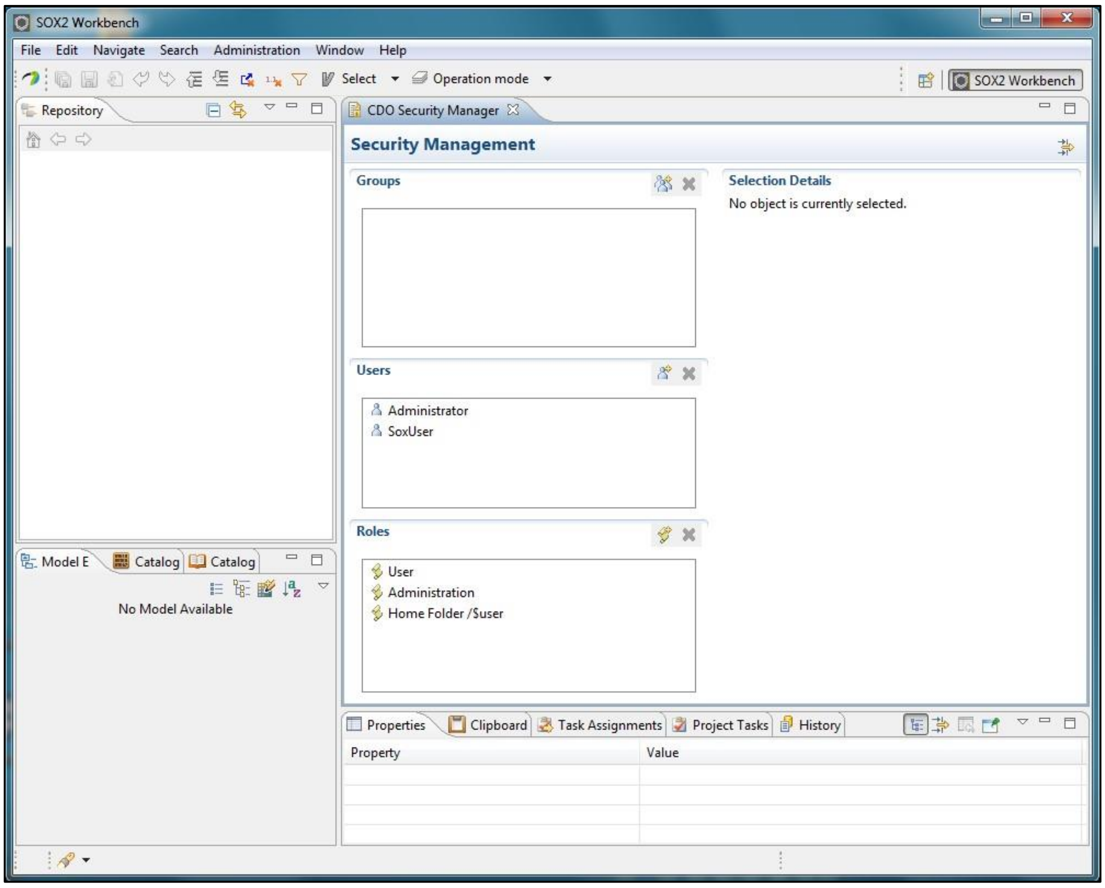
Authenticate and enter your new password.

|  |  |
|---|---|
| User ID: | Administrator |
| Password: |  |
| New Password: |  |
| Repeat Password: |  |

OK Cancel

- "User administration" allows the Administrator to manage all relevant settings concerning groups, users, and roles.





## 6. General Conception

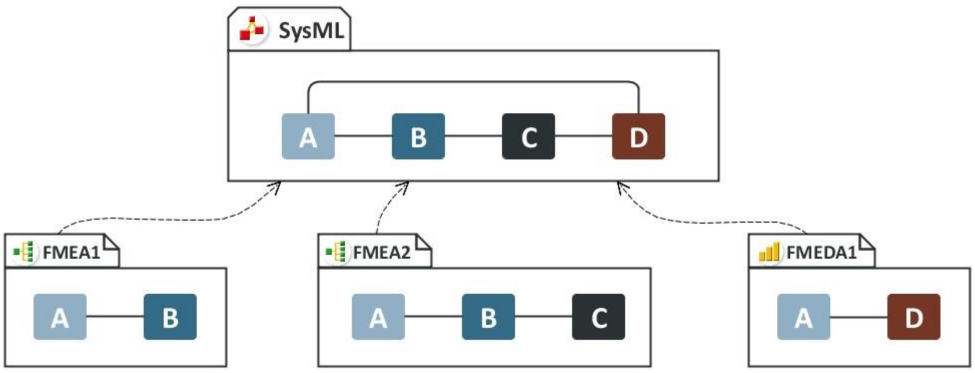
### Short Summary

- The same object (e.g., a requirement, system element or function) can be used in multiple documents and diagrams within a project. Modifying the object in one document affects all occurrences of the object.
- The system design ("Model Explorer" view) contains all objects available in the project. Other documents and diagrams contain only a subset of these objects. Creating (or importing) a new object in a document automatically adds the object to the system design. Deleting an object from a document does not delete the object in the system design unless explicitly done so.
- Exception: The set of SysML requirements available in the system design is exactly the same as the set of requirements in the "Requirements" module.
- The main relationships between objects relevant over different documents (e.g., allocation of a function to a system element) are kept consistent over all documents, too.
- If a change in a document or the system design could affect other documents, a "Refactoring" dialogue appears, which displays the consequences.
- Benefits for the user:
  - New documents can be easily created based on existing data. ○ All documents are always automatically kept consistent.

- All created or imported elements are automatically available in the SysML system design.

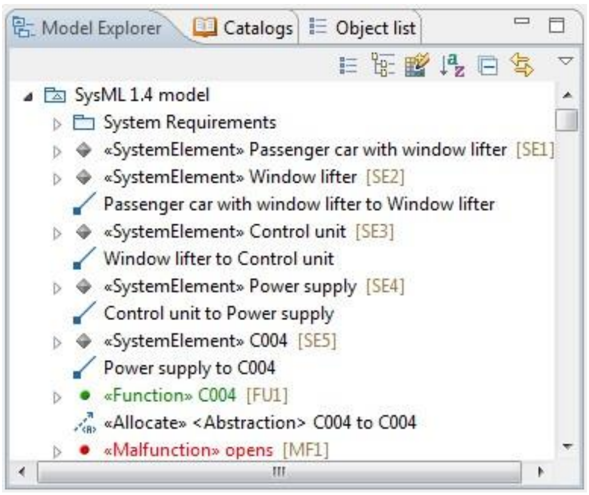
## Details

In SOX v3.0, the same objects can be used in multiple modules and documents. For instance, a system element *SE1* can be defined in the "System Design" module and can then be used in multiple safety analyses, e.g., multiple FMEA documents *FMEA1* and *FMEA2*. There are no copies of SE1: All documents (System Design, FMEA1, and FMEA2) refer to the same object. This means that modifying the object SE1 within one of these documents automatically results in a modification of all other documents that contain the same element.



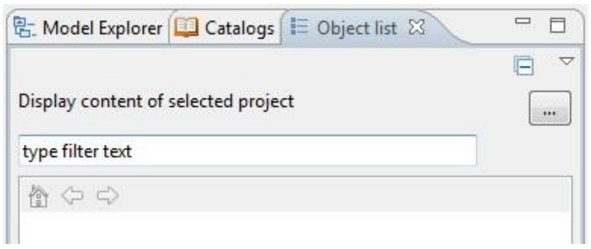
The system design contains all objects to be reused within multiple modules. Those reusable objects are represented by SOX-specific SysML stereotypes in the system design. (A stereotype represents a custom variation of a standard SysML/UML element – for instance, the stereotype *System Element* represents a specific SysML block that is interpreted in SOX as a system element.) The supported reusable objects (represented by stereotypes) are: *system element*, *function* (including subtypes such as safety functions, diagnoses, and process characteristics), *malfunction*, *requirement*, and *safety goal*. An exception are project tasks, which can also be used in multiple documents but are independent from the system design and are managed in the "Project Task" view.

Whenever a new instance of one of these objects is created (e.g., a new system element in an FMEA) it is automatically added to the system design. All existing system design elements are listed in the "Model Explorer" view (see below) and can be reused from there, e.g., by just dragging and dropping them into other documents.

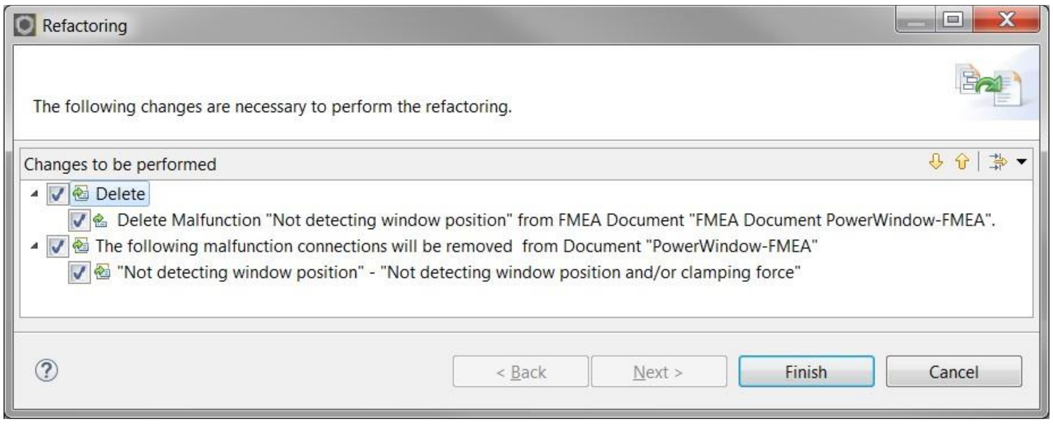


SysML 1.4 model
System Requirements
«SystemElement» Passenger car with window lifter [SE1]
«SystemElement» Window lifter [SE2]
Passenger car with window lifter to Window lifter
«SystemElement» Control unit [SE3]
Window lifter to Control unit
«SystemElement» Power supply [SE4]
Control unit to Power supply
«SystemElement» C004 [SE5]
Power supply to C004
«Function» C004 [FU1]
«Allocate» <Abstraction> C004 to C004
«Malfunction» opens [MF1]

Alternatively, it is possible to open one or more "Object List" views (see below) to show lists of all existing elements of a specific type.



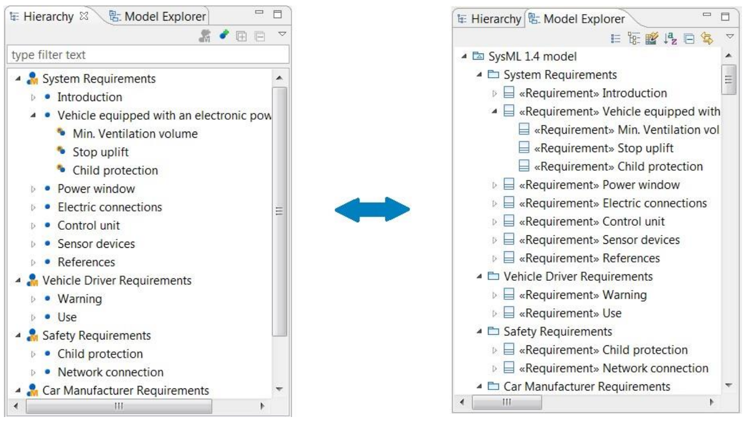
The system design always contains all existing reusable objects (listed in the Model Explorer). But a specific document (e.g., FMEA, FMEDA, FTA document) or diagram (e.g., SysML Block Definition Diagram) contains only a subset of them. For instance, an FMEA document displays only those system elements that are relevant in the context of this specific document. Different documents (e.g., FMEAs) can contain different subsets of elements. As a consequence, creating a new object within an FMEA document automatically adds this object to the system design, but creating a new object within the system design does not automatically add the object to other documents/diagrams (as those contain only subsets). Analogously, deleting an object in the system design deletes the object also in all other documents/diagrams, but deleting an object within another document/ diagram does not automatically result in deletion from the system design. SOX provides a refactoring dialog (see figure below) which prompts information about the consequences of object deletion whenever an object is referred to by other documents.



The following changes are necessary to perform the refactoring.

- ✓ Delete
  - ✓ Delete Malfunction "Not detecting window position" from FMEA Document "FMEA Document PowerWindow-FMEA".
- ✓ The following malfunction connections will be removed from Document "PowerWindow-FMEA"
  - ✓ "Not detecting window position" - "Not detecting window position and/or clamping force"

An exception are requirement documents (RM documents), as there should never be a requirement that exists only in the system design but is not contained in a RM document. Hence, the relationship between requirements in the system design and requirements in RM documents is 1:1, i.e., adding/deleting a requirement on one side automatically results in addition/deletion on the other side. (By default, Requirements created in a RM document are added in the system design into a package with the same name as the RM document, but they can be freely moved within the system design without effects on the RM documents.) This means that imported requirements (e.g., using ReqIF import) become directly available in the system design, e.g., to link them with system design elements.


Hierarchy
Model Explorer
type filter text
System Requirements
Introduction
Vehicle equipped with an electronic pow
Min. Ventilation volume
Stop uplift
Child protection
Power window
Electric connections
Control unit
Sensor devices
References
Vehicle Driver Requirements
Warning
Use
Safety Requirements
Child protection
Network connection
Car Manufacturer Requirements
SysML 1.4 model
System Requirements
«Requirement» Introduction
«Requirement» Vehicle equipped with
«Requirement» Min. Ventilation vol
«Requirement» Stop uplift
«Requirement» Child protection
«Requirement» Power window
«Requirement» Electric connections
«Requirement» Control unit
«Requirement» Sensor devices
«Requirement» References
Vehicle Driver Requirements
«Requirement» Warning
«Requirement» Use
Safety Requirements
«Requirement» Child protection
«Requirement» Network connection
Car Manufacturer Requirements

The following table lists the types of objects that can be used over multiple documents/diagrams (leftmost column) and their representation within a specific document (other columns):

| **Object Type** | **Usage in Document** | | | | |
|---|---|---|---|---|---|
| | **SysML Diagram** | **FMEA Document** | **RE Document** | **FMEDA Document** | **FTA Document** |
| **System Element** | System Element (stereotyped SysML Block) | System Element | Module | Module | – |
| **Function** | Function (stereotyped SysML Block) | Function | – | Hardware Function | – |
| **Malfunction** | Malfunction (stereotyped SysML Block) | Malfunction | – | Hardware Failure | Malfunction |
| **Safety Goal** | Safety Goal (stereotyped SysML Requirement) | Safety Goal | Safety Goal | Safety Goal | – |
| **Requirement** | SysML Requirement | Requirement | Requirement | Requirement | Requirement |

The main relationships between elements are stored in the system design as well and, hence, kept consistent across all documents: Hierarchy of system elements, hierarchy of functions, assignment of functions to system elements, assignment of malfunction to functions, cause-effect relationships between malfunctions, assignment of safety goals to functions, and assignment of requirements to system design elements. Again, adding such a relationship in one document automatically creates a corresponding relationship in the system design (but not vice versa) and deleting such a relationship in the system design can result in appearance of a "Refactoring" dialog that informs about the consequences.

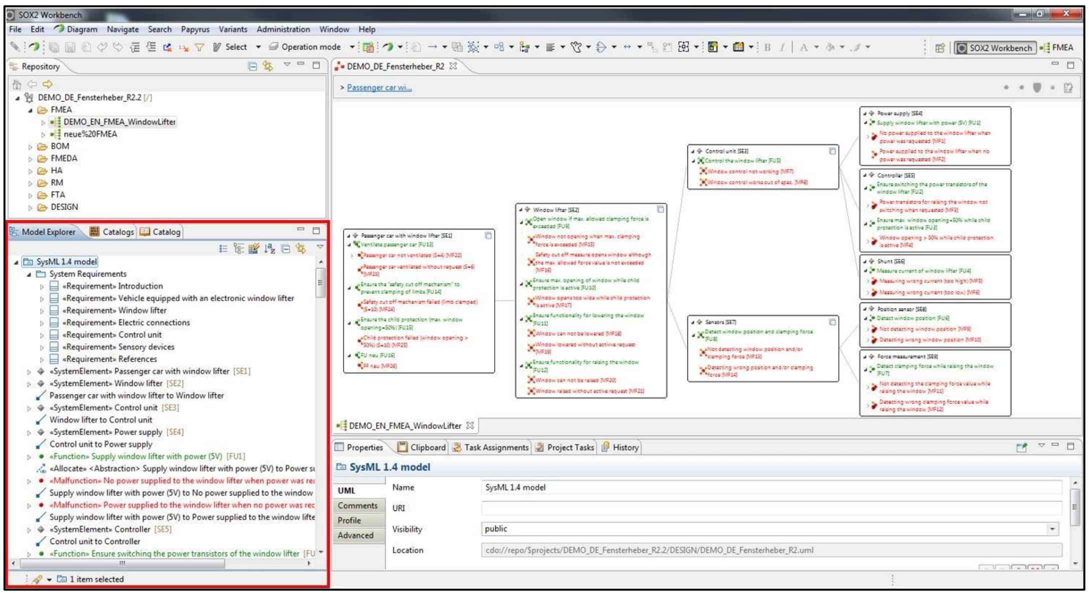
The following table lists the relationships that are relevant in multiple types of documents/diagrams and their meaning within a certain document/diagram type:

| **Relationship** | **Usage in Document** | | | | |
|---|---|---|---|---|---|
| | **SysML Diagram** | **FMEA Document** | **RE Document** | **FMEDA Document** | **FTA Document** |
| Hierarchy of system elements | Composition | Hierarchy of system elements | Hierarchy of modules | Hierarchy of modules | – |
| Hierarchy of functions | Composition | Connections in function net | – | – | – |
| Assignment of functions to system elements | Allocate relationship | Assignment of functions to system elements | – | Available functions of modules | – |
| Assignment of malfunctions to functions | Composition | Assignment of malfunctions to functions | – | Available malfunctions of functions | – |
| Cause-effect relationships of malfunctions | "Effects" relationship | Connections in failure net | – | – | – |
| Assignment of safety goals to (mal)functions | Satisfy relationship | Assignment of safety goals to (mal)functions | – | System effect safety goals | – |
| Assignment of malfunctions to requirements | Derive relationship | Assignment of malfunctions to requirements | – | – | – |

| | | | | | |
|---|---|---|---|---|---|
| Assignment of malfunctions to safety goals. | Derive relationship | Assignment of malfunctions to requirements | – | – | – |
| Assignment of requirements to any other object | Satisfy relationship | Assignment of requirement | Assignment of requirement | Assignment of requirement | Assignment of requirement |
| Decomposition of safety requirements | Decompose relationship | Decomposition | Decomposition | Decomposition | Decomposition |

## 7. System Design and SOX-specific SysML Profile

- SOX supports system design with SysML 1.4.
- Created/imported objects that can be reused over multiple documents (requirements, functions, malfunctions, diagnoses, and safety goals) and relationships between them are available in the Model Explorer as stereotyped SysML elements (see Section 6).

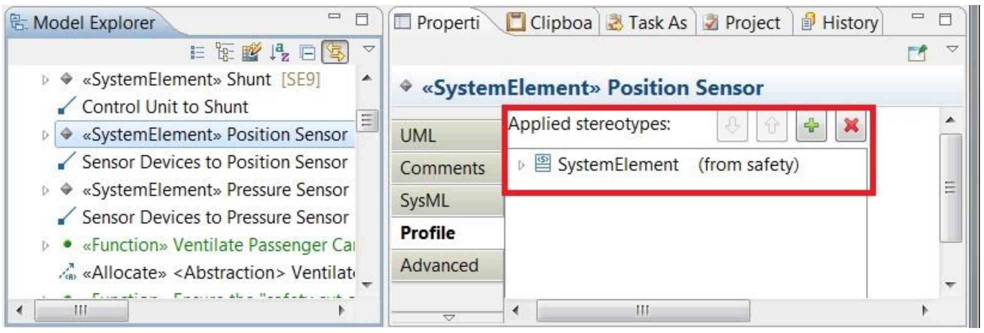


SysML 1.4 model
System Requirements
«Requirement» Introduction
«Requirement» Vehicle equipped with an electronic window lifter
«Requirement» Window lifter
«Requirement» Electric connections
«Requirement» Control unit
«Requirement» Sensory devices
«Requirement» References
«SystemElement» Passenger car with window lifter [SE1]
«SystemElement» Window lifter [SE2]
Passenger car with window lifter to Window lifter
«SystemElement» Control unit [SE3]
Window lifter to Control unit
«SystemElement» Power supply [SE4]
Control unit to Power supply
«Function» Supply window lifter with power (5V) [FU1]
«Allocate» <Abstraction> Supply window lifter with power (5V) to Power su
«Malfunction» No power supplied to the window lifter when power was rec
Supply window lifter with power (5V) to No power supplied to the window
«Malfunction» Power supplied to the window lifter when no power was rec
Supply window lifter with power (5V) to Power supplied to the window lifte
«SystemElement» Controller [SE5]
Control unit to Controller
«Function» Ensure switching the power transistors of the window lifter [FU
1 item selected

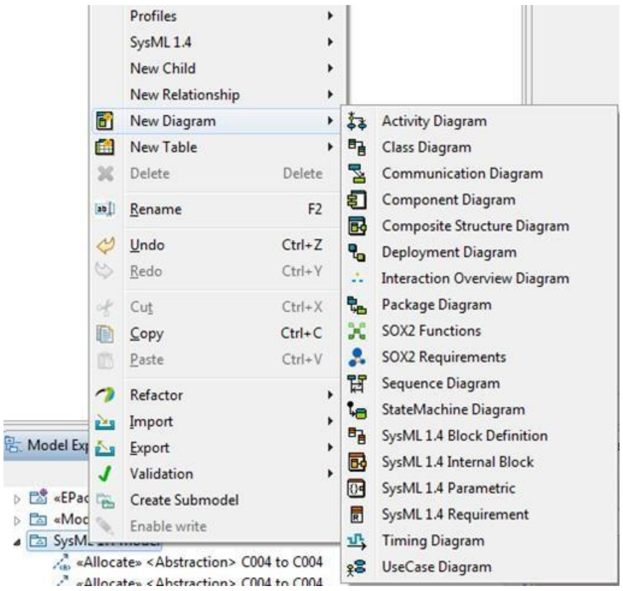
SysML 1.4 model

| | | |
|---|---|---|
| UML | Name | SysML 1.4 model |
| Comments | URI | |
| Profile | Visibility | public |
| Advanced | Location | cdo://repo/$projects/DEMO_DE_Fensterheber_R2.2/DESIGN/DEMO_DE_Fensterheber_R2.uml |

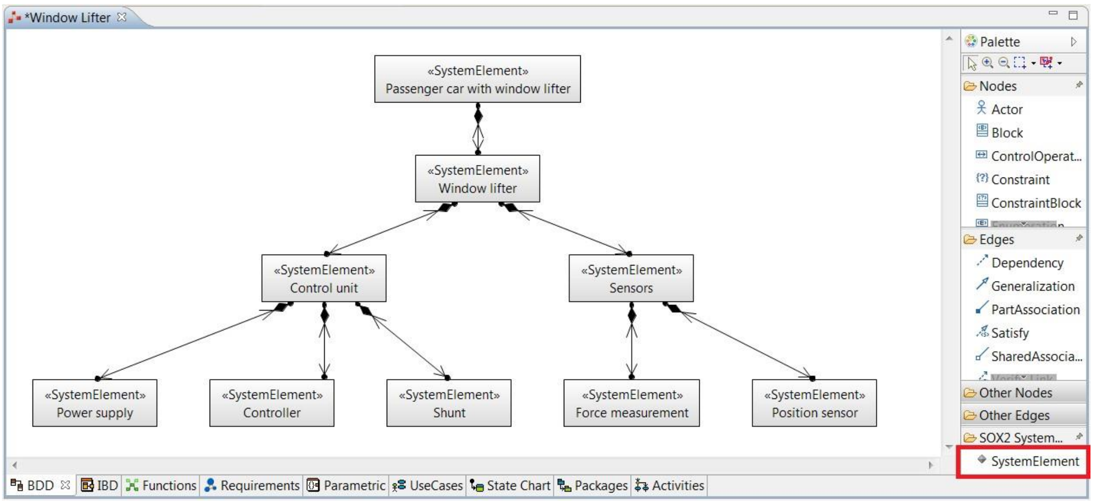
- In SysML, stereotypes can be applied to SysML elements to mark a specific meaning of the element. SOX uses the following stereotypes: *SystemElement*, *Function* (including subtypes such as *SafetyFunction*, *Diagnosis* or *ProcessCharacteristics*), *Malfunction*, *Requirement*, *SafetyGoal,* and *Effect* (indicates a cause-effect relationship between malfunctions).
- It is possible to modify the applied stereotype of an existing SysML element in the “Properties” view. However, note that <u>applying/removing stereotypes changes the meaning of an element which can have unintended consequences</u>, in particular, if the element is already used in other documents.


Model Explorer
«SystemElement» Shunt [SE9]
Control Unit to Shunt
«SystemElement» Position Sensor
Sensor Devices to Position Sensor
«SystemElement» Pressure Sensor
Sensor Devices to Pressure Sensor
«Function» Ventilate Passenger Ca
«Allocate» <Abstraction> Ventilat
Properti
Clipboa
Task As
Project
History
«SystemElement» Position Sensor
UML
Comments
SysML
Profile
Advanced
Applied stereotypes:
SystemElement (from safety)

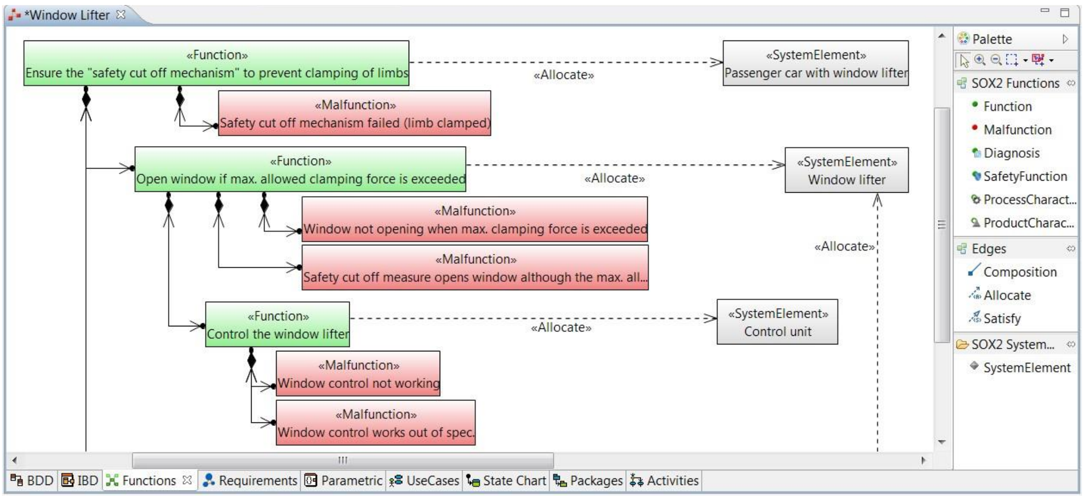
- Create all kinds of SysML and UML diagrams in order to define a complete system design (rightclick on intended container, e.g., the top-level package “SysML 1.4 model”). In addition, SOX provides two additional diagram types, SOX Functions diagram and SOX Requirements diagram, to be used to create SOX-specific stereotypes in a convenient manner.



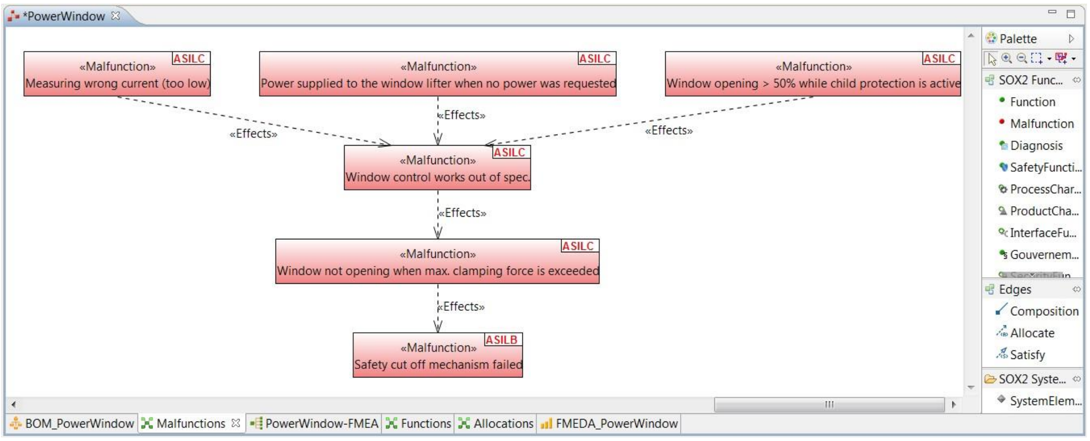
- Create a *SysML 1.4 Block Definition* diagram to specify the system structure. Create diagram content by selecting a tool in the palette and clicking at the position in the diagram where to place the element. Note that blocks that should be interpreted as system elements by SOX must have the stereotype *SystemElement* applied. Use the tool on the bottom of the palette (see figure) to easily create system elements. The hierarchy of system elements is defined by a composition relationship (“PartAssociation” tool) from the parent to the child system element.



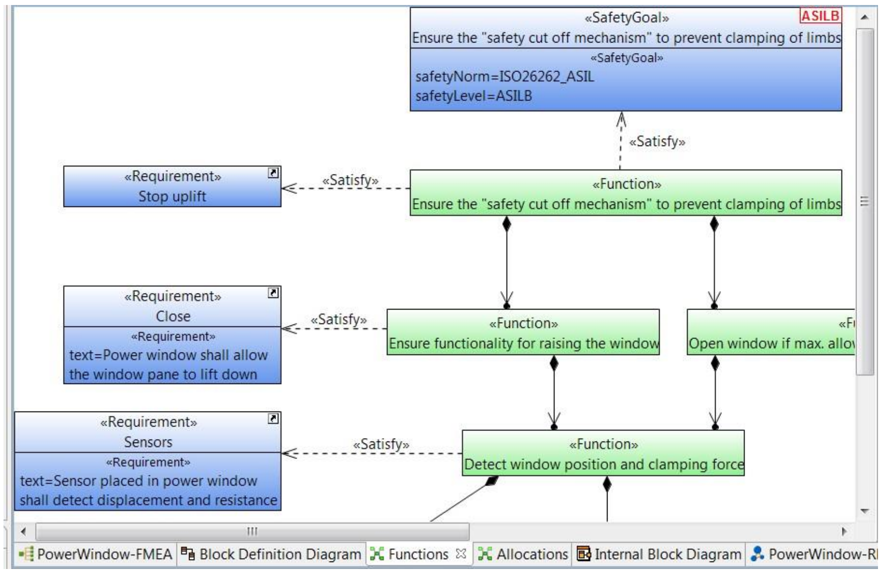
- Create a SOX Function diagram to specify a function, their hierarchy, associated malfunctions, and allocations of functions to system elements. Malfunctions are assigned to functions via a *Composition* relationship from the function to the malfunction. Functions are assigned to system elements via an *Allocate* relationship.


*Window Lifter
«Function» Ensure the "safety cut off mechanism" to prevent clamping of limbs
«Allocate»
«SystemElement» Passenger car with window lifter
«Malfunction» Safety cut off mechanism failed (limb clamped)
«Function» Open window if max. allowed clamping force is exceeded
«Allocate»
«SystemElement» Window lifter
«Malfunction» Window not opening when max. clamping force is exceeded
«Malfunction» Safety cut off measure opens window although the max. all...
«Allocate»
«Function» Control the window lifter
«Allocate»
«SystemElement» Control unit
«Malfunction» Window control not working
«Malfunction» Window control works out of spec.
Palette
SOX2 Functions
Function
Malfunction
Diagnosis
SafetyFunction
ProcessCharact...
ProductCharac...
Edges
Composition
Allocate
Satisfy
SOX2 System...
SystemElement
BDD
IBD
Functions
Requirements
Parametric
UseCases
State Chart
Packages
Activities

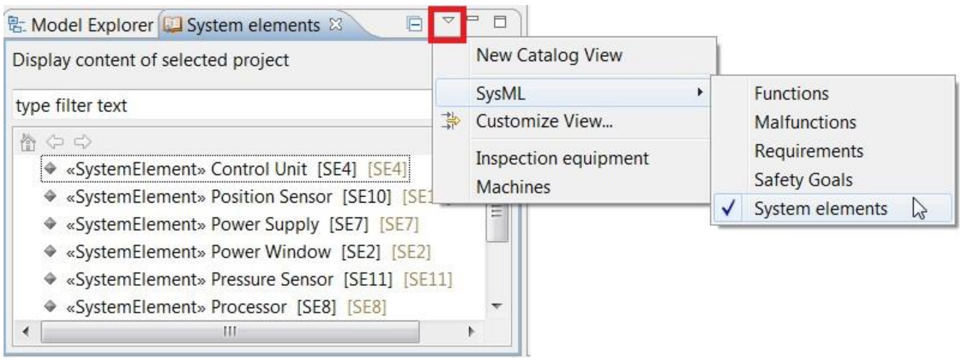
- Connections between malfunctions defined in failure nets (cause-effect-relationships) are added to the system design as well. They are represented in SysML by dependencies with a SOX-specific stereotype *Effects* applied. Hence, the failure net connections can be represented by, e.g., a Function diagram.


*PowerWindow
«Malfunction» Measuring wrong current (too low) ASILC
«Malfunction» Power supplied to the window lifter when no power was requested ASILC
«Malfunction» Window opening > 50% while child protection is active ASILC
«Effects»
«Effects»
«Effects»
«Malfunction» Window control works out of spec. ASILC
«Effects»
«Malfunction» Window not opening when max. clamping force is exceeded ASILC
«Effects»
«Malfunction» Safety cut off mechanism failed ASILB
Palette
SOX2 Func...
Function
Malfunction
Diagnosis
SafetyFuncti...
ProcessChar...
ProductCha...
InterfaceFu...
Gouvernem...
Edges
Composition
Allocate
Satisfy
SOX2 Syste...
SystemElem...
BOM_PowerWindow
Malfunctions
PowerWindow-FMEA
Functions
Allocations
FMEDA_PowerWindow

- Requirements and Safety Goals can be created in a SOX Requirements diagram. However, it is more convenient to create them using the dedicated *Requirements Editor* (in the "Requirements" module) and the "Safety Goal" *View* and then just drag them from the Model Explorer into SysML diagrams where needed. Requirements and Safety Goals are assigned to system elements and functions via a *Satisfy* relationship.

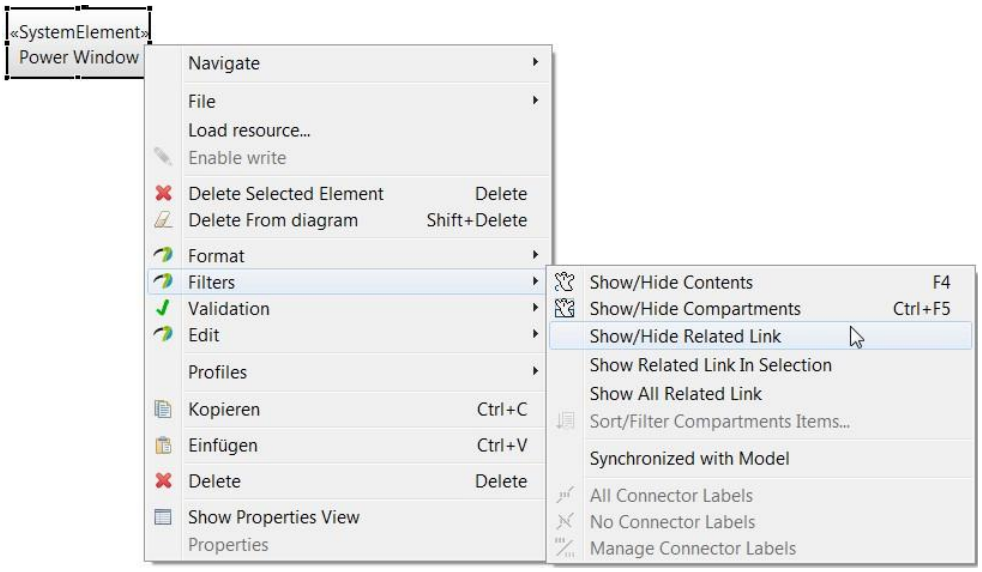

«SafetyGoal»
ASILB
Ensure the "safety cut off mechanism" to prevent clamping of limbs
«SafetyGoal»
safetyNorm=ISO26262_ASIL
safetyLevel=ASILB
«Satisfy»
«Requirement»
Stop uplift
«Satisfy»
«Function»
Ensure the "safety cut off mechanism" to prevent clamping of limbs
«Requirement»
Close
«Requirement»
text=Power window shall allow the window pane to lift down
«Satisfy»
«Function»
Ensure functionality for raising the window
Open window if max. allo
«Requirement»
Sensors
«Requirement»
text=Sensor placed in power window shall detect displacement and resistance
«Satisfy»
«Function»
Detect window position and clamping force
PowerWindow-FMEA
Block Definition Diagram
Functions
Allocations
Internal Block Diagram
PowerWindow-R

- As usual in SysML, it is up to the user what type and how many diagrams to create. Diagrams are just a specific view on the overall system design defined in the Model Explorer (see Section 6). For instance, allocations of functions to system elements can be defined in a BDD instead of/in addition to a SOX Functions diagram.
- Objects defined in other SOX documents (e.g., system elements created in an FMEA document) are automatically added to the system design. Hence, they are already available in the Model Explorer and do not need to be created again in SysML. To add them to a SysML diagram, just drag them from the Model Explorer or the Object List into the diagrams. The Object List supports selection of an object type (e.g., system element) and to filter by typing a filter text.

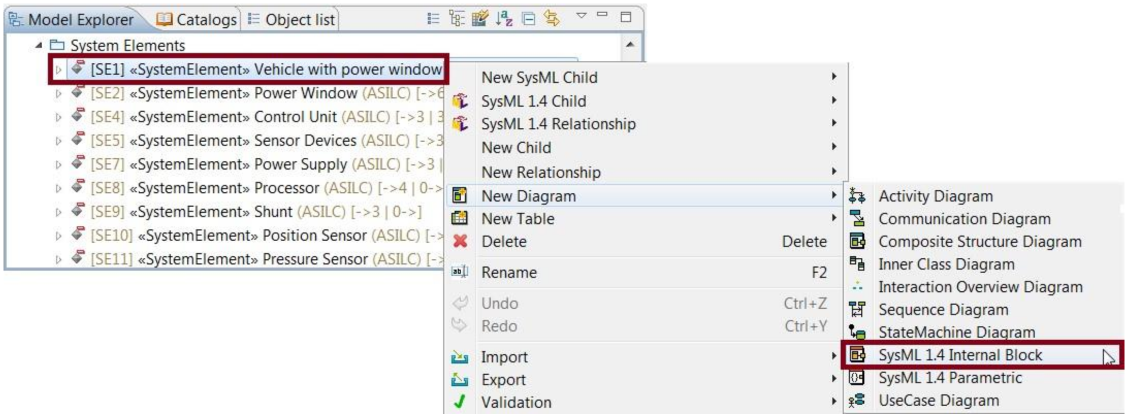


«SystemElement» Control Unit [SE4] [SE4]
«SystemElement» Position Sensor [SE10] [SE1
«SystemElement» Power Supply [SE7] [SE7]
«SystemElement» Power Window [SE2] [SE2]
«SystemElement» Pressure Sensor [SE11] [SE11]
«SystemElement» Processor [SE8] [SE8]

To add existing relationships (e.g. hierarchy of system elements as defined in an FMEA) to a SysML diagram, either drag the relationships from the Model Explorer into the diagram or use the “Show/Hide Related Link” tool in a diagram.



- Note that since Beta 3, SysML Internal Block Diagrams and Parametric Diagrams must be created with a single System Element as root element (complaint to the SysML standard). Hence, Internal Block Diagrams and Parametric Diagrams should be created directly on a System Element by right-click on the System Element in the Model Explorer.


Model Explorer
Catalogs
Object list
System Elements
[SE1] «SystemElement» Vehicle with power window
[SE2] «SystemElement» Power Window (ASILC) [->6
[SE4] «SystemElement» Control Unit (ASILC) [->3 | 3
[SE5] «SystemElement» Sensor Devices (ASILC) [->3
[SE7] «SystemElement» Power Supply (ASILC) [->3 |
[SE8] «SystemElement» Processor (ASILC) [->4 | 0->
[SE9] «SystemElement» Shunt (ASILC) [->3 | 0->]
[SE10] «SystemElement» Position Sensor (ASILC) [->
[SE11] «SystemElement» Pressure Sensor (ASILC) [->
New SysML Child
SysML 1.4 Child
SysML 1.4 Relationship
New Child
New Relationship
New Diagram
New Table
Delete Delete
Rename F2
Undo Ctrl+Z
Redo Ctrl+Y
Import
Export
Validation
Activity Diagram
Communication Diagram
Composite Structure Diagram
Inner Class Diagram
Interaction Overview Diagram
Sequence Diagram
StateMachine Diagram
SysML 1.4 Internal Block
SysML 1.4 Parametric
UseCase Diagram

  - **Note that creation of an Internal Block Diagram or Parametric Diagram directly on a package (instead of a System Element) results in creation of a new Block to be used as the diagram's root element. You then need to assign the stereotype "System Element" manually to this Block.** To manually assign a stereotype, select the element and go to the tab "Profile" in the Properties View (see SOX help).
- Further information about the diagram tools can be found in the SysML help in SOX

## 8. Modeling Functions Using Activities

In SOX, Functions are represented in SysML by UML Classes with the stereotype "Function" applied. However, it is possible to use UML Activities instead of UML Classes. For this, the stereotype "Function" can be applied to Activities.

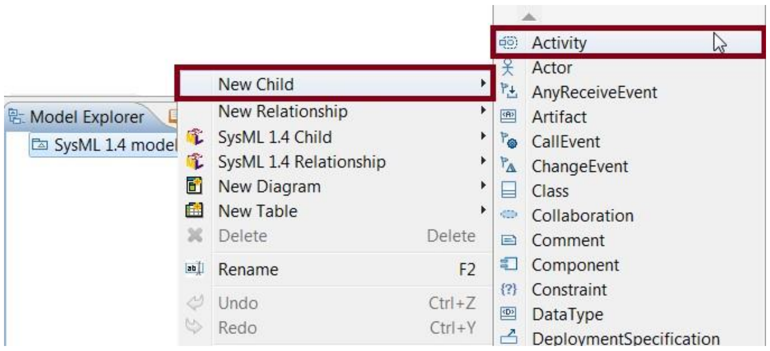
When creating SysML models using Activities for representing Functions, please note the following:

- When creating Functions via the diagram palette, a Function based on a UML Class is created (not a UML Activity).
- When creating Functions in other documents than SysML (e.g., FMEA documents), a Function based on a UML Class is created (not a UML Activity).
- The SOX v3.0 provides additional support for representing Functions by UML Activities.

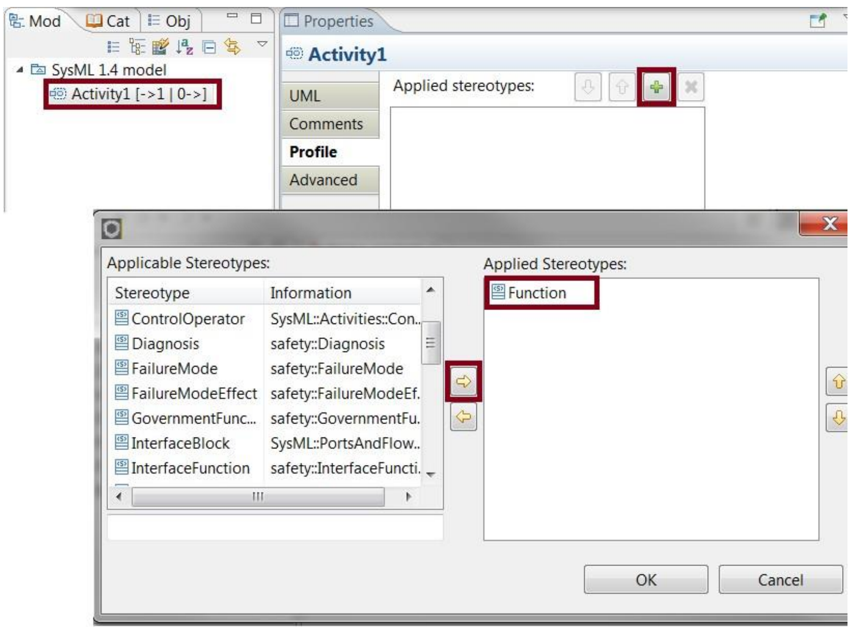
The remainder of this section explains how Activities and Activity Diagrams can be used.

### Creating Activities that Represent Functions

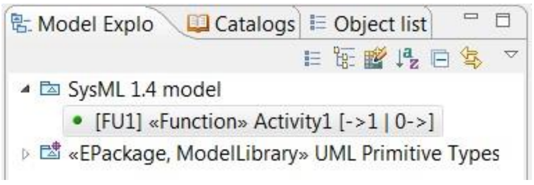
- Create a UML Activity in the Model Explorer by right-click on the desired package > New Child > Activity



- Assign the stereotype "Function" to the Activity:
  - Make sure the activity is selected in the Model Explorer
  - Go to the tab "Profile" in the Properties View to manage stereotypes for the selected Activity. To assign a stereotype, click on the "+" button.
  - A dialog appears where to select the stereotypes to apply. Select the stereotype "Function" and apply it using the arrow button. Finally, press "Ok".



- The activity is now stereotyped as a “Function”.

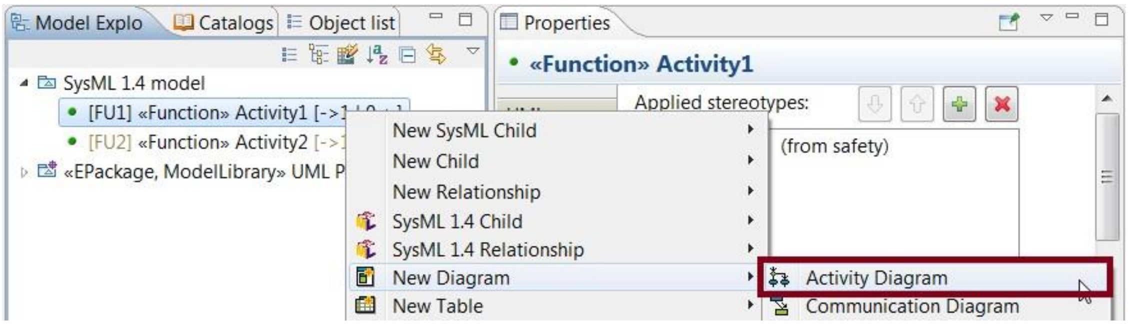

Model Explo
Catalogs
Object list
SysML 1.4 model
[FU1] «Function» Activity1 [->1 | 0->]
«EPackage, ModelLibrary» UML Primitive Types

- Functions represented by stereotyped Activities can be used in analogous way as Functions represented by stereotyped Classes; see Section 7. Drag them from the model explorer into diagrams (e.g., SOX Function Diagram) to:
  - Specify the hierarchy of Functions, ○ Add Malfunctions, ○ Allocate Functions to System Elements, ○ Assign Safety Goals or Safety Requirements.
  - The resulting models can be used for safety analysis, e.g., to automatically derive an FMEA document from the system design

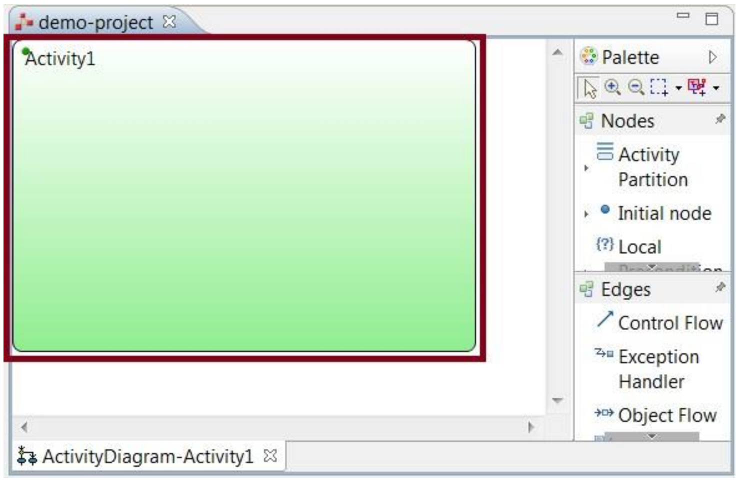
## Create Activity Diagrams to Specify the Behavior of Functions

An Activity Diagram can be used to specify the detailed behavior of a Function (i.e., stereotyped Activity) including calls of other Functions (Activities).

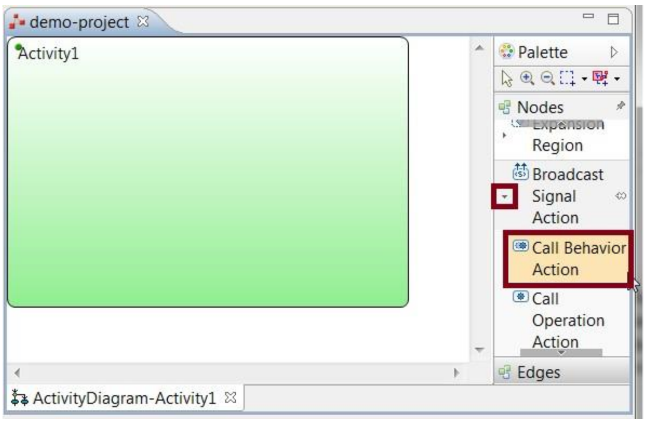
- Select the Activity that should become the root element of your Activity Diagram in the Model Explorer. Right-click on the activity and select New Diagram > Activity Diagram



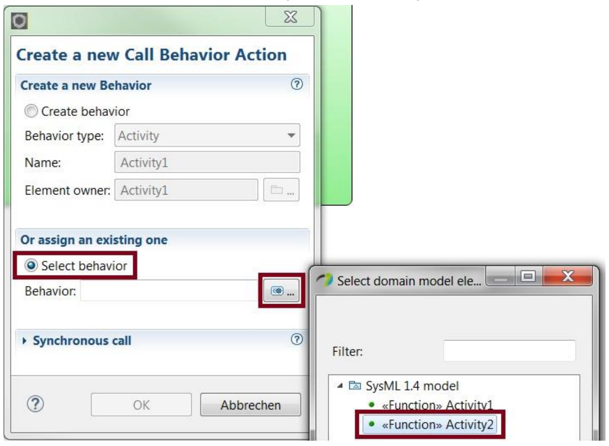
- After specifying a name, an Activity diagram appears with the selected activity as root node. Any content (Actions, Object Flows) can be added from the palette in the usual way. **Note that any content must be placed <u>inside</u> the diagram's root Activity**. The activity node can be enlarged by dragging its corners with the mouse.



- In UML/SysML, calling another activity (i.e., sub-function) is modelled by a Call Behavior Action. Due to the large number of possible elements in UML activity diagrams, related elements are grouped together in the palette. You can find the Call Behavior Action below the Broadcast Signal Action in the palette (expand the Broadcast Signal Action by clicking on the little triangle on its left).



- When creating a new Call Behavior Action, a dialogue window appears asking to select the behavior to be called by this Call Behavior Action. Select an existing Function (i.e. another Activity) using the option "Or assign an existing one" at the dialogue window's bottom part. (When creating a new Activity instead, don't forget to apply the stereotype "Function" afterwards if the Activity should represent a Function)


**Create a new Call Behavior Action**

**Create a new Behavior**

Create behavior

Behavior type: Activity

Name: Activity1

Element owner: Activity1

**Or assign an existing one**

Select behavior

Behavior:

**Synchronous call**

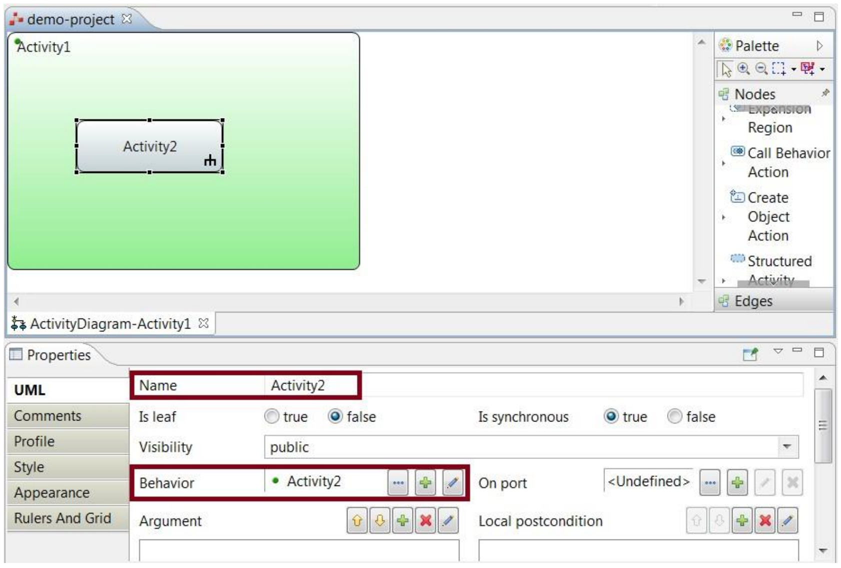
OK | Abbrechen

Select domain model ele...

Filter:

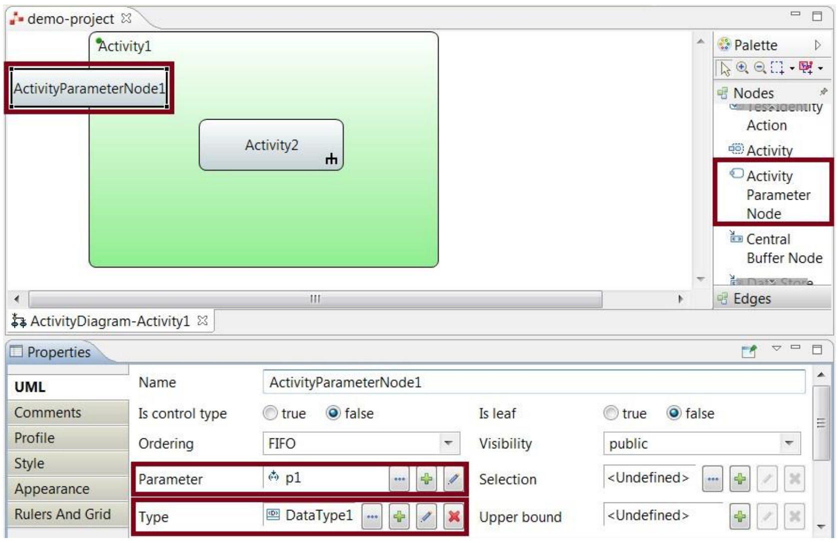
SysML 1.4 model
- «Function» Activity1
- «Function» Activity2

- The behavior called by this Call Behavior Action is visible in the Properties View as property "Behavior" (and can be edited there). Note that only the Call Behavior Action's name is shown on the label in the diagram (not the name of the called behavior), so it is useful to choose a name that indicates the called behavior.



| | | | |
|---|---|---|---|
| Name | Activity2 | | |
| Is leaf | true [✓ false] | Is synchronous | [✓ true] false |
| Visibility | public | | |
| Behavior | Activity2 | On port | <Undefined> |
| Argument | | Local postcondition | |

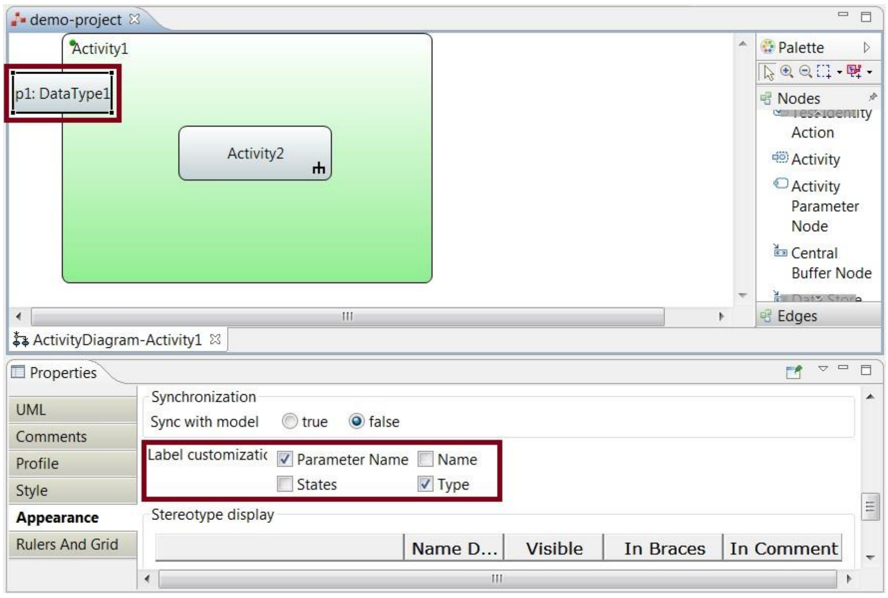
- Extend the Activity Diagram by adding, e.g., Actions, Activity Parameter Nodes, Input Pins, Output Pins, Control Flows, etc. The example below shows creation of an Activity Parameter Node. You can specify its Parameter and its type in the Properties View. For instance, create a new Data Type as its type or select an existing one.



| | | | |
|---|---|---|---|
| Name | ActivityParameterNode1 | | |
| Is control type | true [✓ false] | Is leaf | true [✓ false] |
| Ordering | FIFO | Visibility | public |
| Parameter | p1 | Selection | <Undefined> |
| Type | DataType1 | Upper bound | <Undefined> |

- In general, labels of diagram elements can be customized via the Tab "Appearance" in the Properties View. For instance, an Activity Parameter Node can display the parameter name and type. This capability is not fully supported for all types of elements.

- In SysML Activity Diagrams, it is optionally possible to visually indicate which Actions (i.e., here the called Functions) are allocated to which System Elements. For this, create an Activity Partition in the Activity Diagram and drag Actions into it.



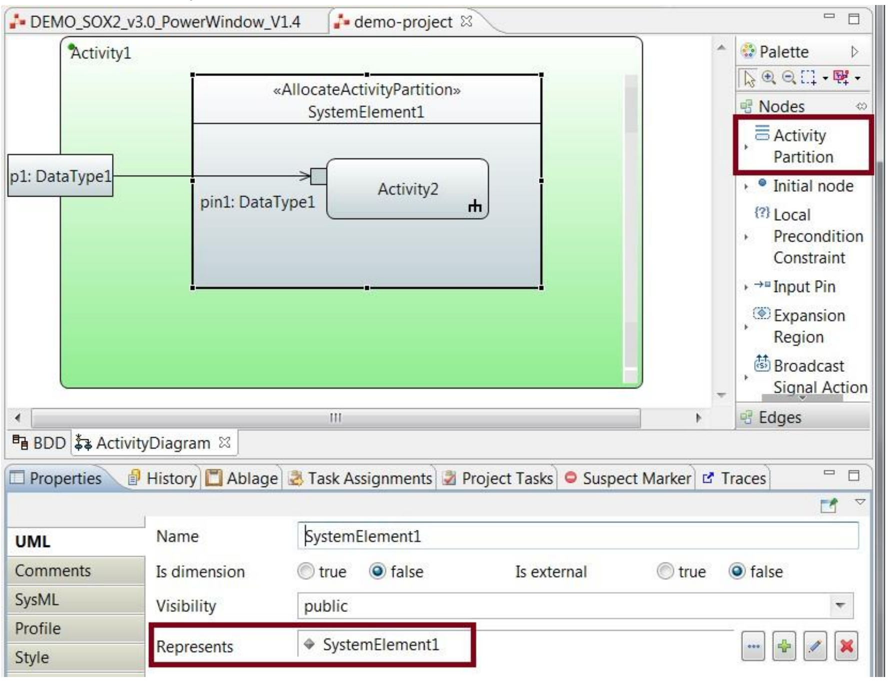
Synchronization
Sync with model ( ) true [✓ false]

Label customizatic [✓ Parameter Name] [ ] Name [ ] States [✓ Type]

Stereotype display

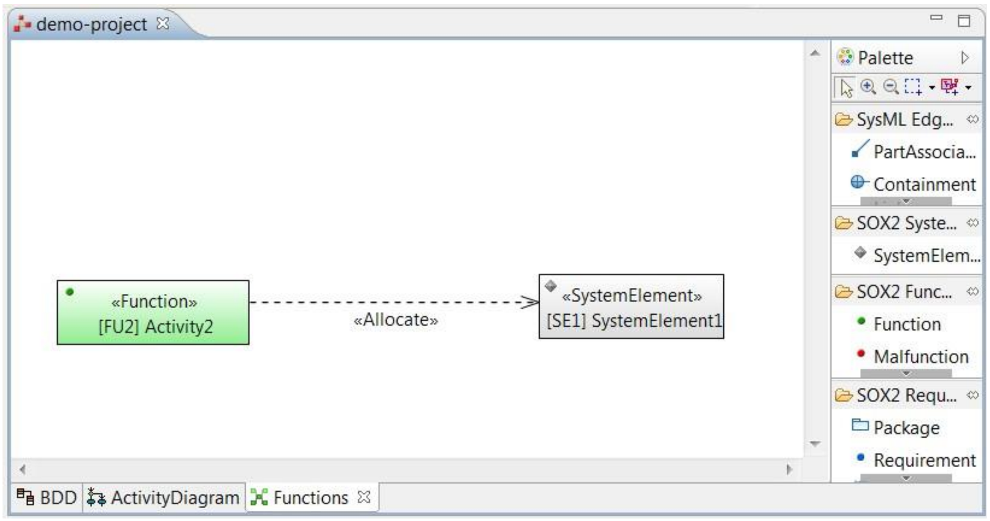
| | Name D... | Visible | In Braces | In Comment |
|---|---|---|---|---|

- To be compliant to the SysML standard, the stereotype “AllocateActivityPartition” can be optionally applied to the Activity Partition. Applying stereotypes is performed using the “Profile” tab in the Properties view (see applying a stereotype to an Activity above).
- An Activity Partition has a property “Represents” (see screenshot). Choose the System Element to be represented by this Activity Partition (i.e., the System Element which the Functions are allocated to). For instance in the example below, the Function “Activity2” is allocated to the System Element “SystemElement1”. Note that an Activity Partition’s label shows only its name.

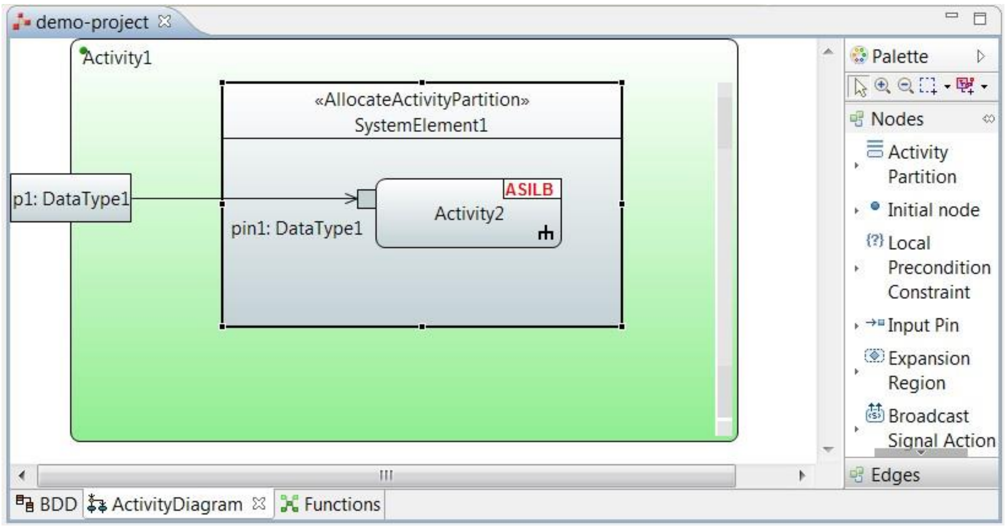


Name: SystemElement1
Is dimension ( ) true [✓ false] Is external ( ) true [✓ false]
Visibility: public
Represents: SystemElement1

- Note that safety analyses (e.g., FMEA) in SOX only consider Allocate relationships that are explicitly defined between Functions and System Elements (e.g., in the SOX Function Diagram as shown below). “AllocateActivityPartitions” are only for the purpose of visualization.



- If a Function inherits a safety level (e.g., from a safety goal, safety requirement, or a related function or malfunction), the safety level is also automatically displayed Activity Diagrams at all Call Behavior Actions that call this Function.



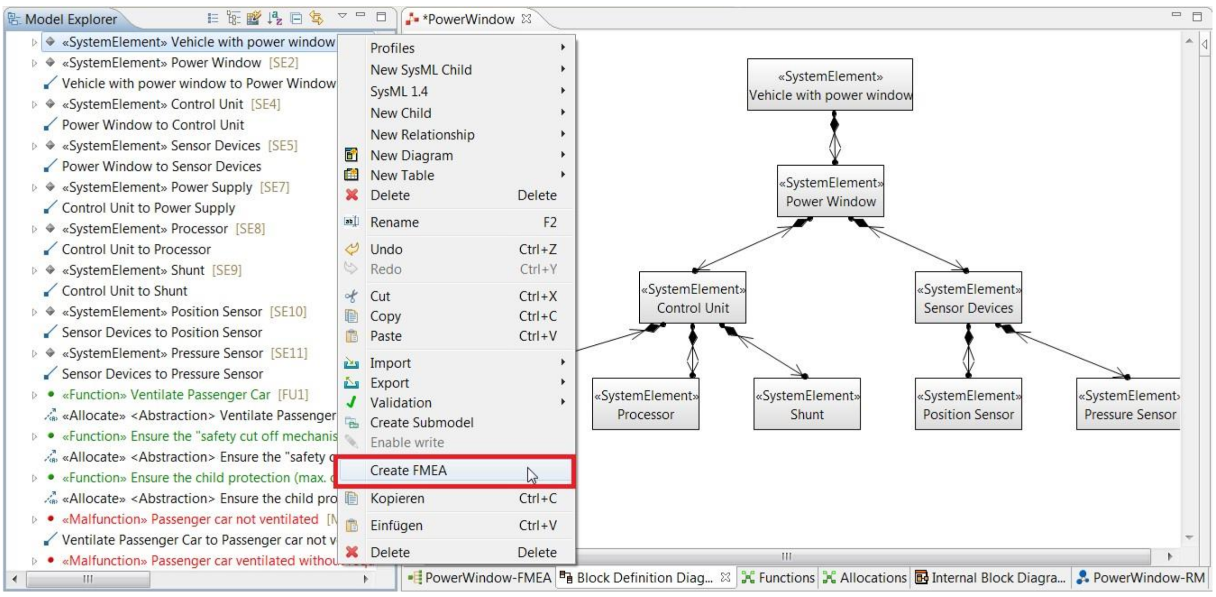
# 9. Linkage between Modules

## System Structure in FMEA and System Design

- The system structure (hierarchy of system elements, allocated functions and malfunctions) created in an FMEA document is automatically added to the system design (see Section 6).
- To visualize the system structure in a SysML diagram, just drag system elements (e.g., from "Model Explorer" view or "Object List" view) into the diagram. To show the relationships in a diagram, either just drag them into the diagram as well or use the ""Show/Hide Related Link" tool in the diagram (see Section 7).
- To use existing system elements, functions and malfunctions in an FMEA document, just drag them to the corresponding location in the FMEA document.
- It is also possible to automatically initialize a new FMEA document with the structure from the system design starting from a chosen root system element. This is described in the next section.

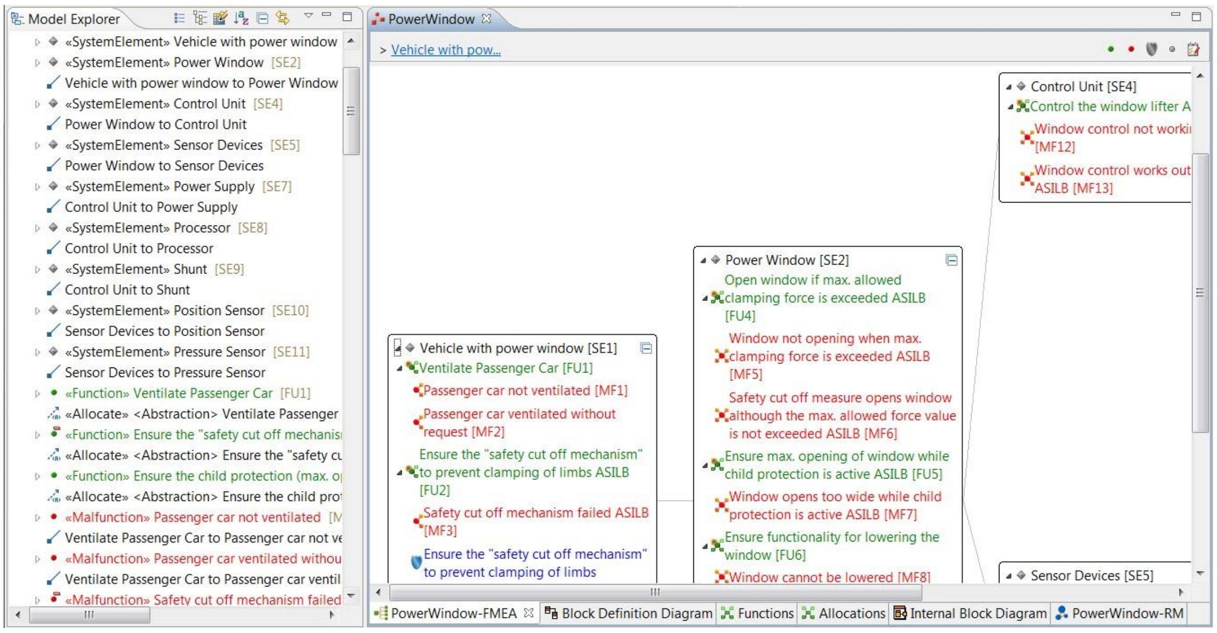
## Automatically Derive an FMEA from System Design

- Once the system design has been defined it can be used to automatically initialize new FMEA documents with the corresponding system structure.
- Select in the Model Explorer the system element that should become the FMEA's root element and click "Create FMEA".



Model Explorer
«SystemElement» Vehicle with power window
«SystemElement» Power Window [SE2]
Vehicle with power window to Power Window
«SystemElement» Control Unit [SE4]
Power Window to Control Unit
«SystemElement» Sensor Devices [SE5]
Power Window to Sensor Devices
«SystemElement» Power Supply [SE7]
Control Unit to Power Supply
«SystemElement» Processor [SE8]
Control Unit to Processor
«SystemElement» Shunt [SE9]
Control Unit to Shunt
«SystemElement» Position Sensor [SE10]
Sensor Devices to Position Sensor
«SystemElement» Pressure Sensor [SE11]
Sensor Devices to Pressure Sensor
«Function» Ventilate Passenger Car [FU1]
«Allocate» <Abstraction> Ventilate Passenger
«Function» Ensure the "safety cut off mechanis
«Allocate» <Abstraction> Ensure the "safety
«Function» Ensure the child protection (max.
«Allocate» <Abstraction> Ensure the child pro
«Malfunction» Passenger car not ventilated
Ventilate Passenger Car to Passenger car not v
«Malfunction» Passenger car ventilated withou
Profiles
New SysML Child
SysML 1.4
New Child
New Relationship
New Diagram
New Table
Delete Delete
Rename F2
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Import
Export
Validation
Create Submodel
Enable write
Create FMEA
Kopieren Ctrl+C
Einfügen Ctrl+V
Delete Delete
*PowerWindow
«SystemElement» Vehicle with power window
«SystemElement» Power Window
«SystemElement» Control Unit
«SystemElement» Sensor Devices
«SystemElement» Processor
«SystemElement» Shunt
«SystemElement» Position Sensor
«SystemElement» Pressure Sensor
PowerWindow-FMEA
Block Definition Diag...
Functions
Allocations
Internal Block Diagra...
PowerWindow-RM

- After selecting a filename and folder for the FMEA, a new FMEA document is created.
- The document is initialized with the chosen system element and all its child system elements (as defined by *Composition* relationships in SysML), all related functions (*Allocate* relationships between system elements and functions), all related malfunctions (*Composition* relationship between functions and malfunctions), and the related safety goals (*Satisfy* relationship between functions and safety goals).



Model Explorer

- «SystemElement» Vehicle with power window
- «SystemElement» Power Window [SE2]
- Vehicle with power window to Power Window
- «SystemElement» Control Unit [SE4]
- Power Window to Control Unit
- «SystemElement» Sensor Devices [SE5]
- Power Window to Sensor Devices
- «SystemElement» Power Supply [SE7]
- Control Unit to Power Supply
- «SystemElement» Processor [SE8]
- Control Unit to Processor
- «SystemElement» Shunt [SE9]
- Control Unit to Shunt
- «SystemElement» Position Sensor [SE10]
- Sensor Devices to Position Sensor
- «SystemElement» Pressure Sensor [SE11]
- Sensor Devices to Pressure Sensor
- «Function» Ventilate Passenger Car [FU1]
- «Allocate» <Abstraction> Ventilate Passenger
- «Function» Ensure the "safety cut off mechanis
- «Allocate» <Abstraction> Ensure the "safety cu
- «Function» Ensure the child protection (max. o
- «Allocate» <Abstraction> Ensure the child prot
- «Malfunction» Passenger car not ventilated [M
- Ventilate Passenger Car to Passenger car not ve
- «Malfunction» Passenger car ventilated withou
- Ventilate Passenger Car to Passenger car ventil
- «Malfunction» Safety cut off mechanism failed

PowerWindow

> Vehicle with pow...

Vehicle with power window [SE1]
- Ventilate Passenger Car [FU1]
  - Passenger car not ventilated [MF1]
  - Passenger car ventilated without request [MF2]
- Ensure the "safety cut off mechanism" to prevent clamping of limbs ASILB [FU2]
  - Safety cut off mechanism failed ASILB [MF3]
  - Ensure the "safety cut off mechanism" to prevent clamping of limbs

Power Window [SE2]
- Open window if max. allowed clamping force is exceeded ASILB [FU4]
  - Window not opening when max. clamping force is exceeded ASILB [MF5]
  - Safety cut off measure opens window although the max. allowed force value is not exceeded ASILB [MF6]
- Ensure max. opening of window while child protection is active ASILB [FU5]
  - Window opens too wide while child protection is active ASILB [MF7]
- Ensure functionality for lowering the window [FU6]
  - Window cannot be lowered [MF8]

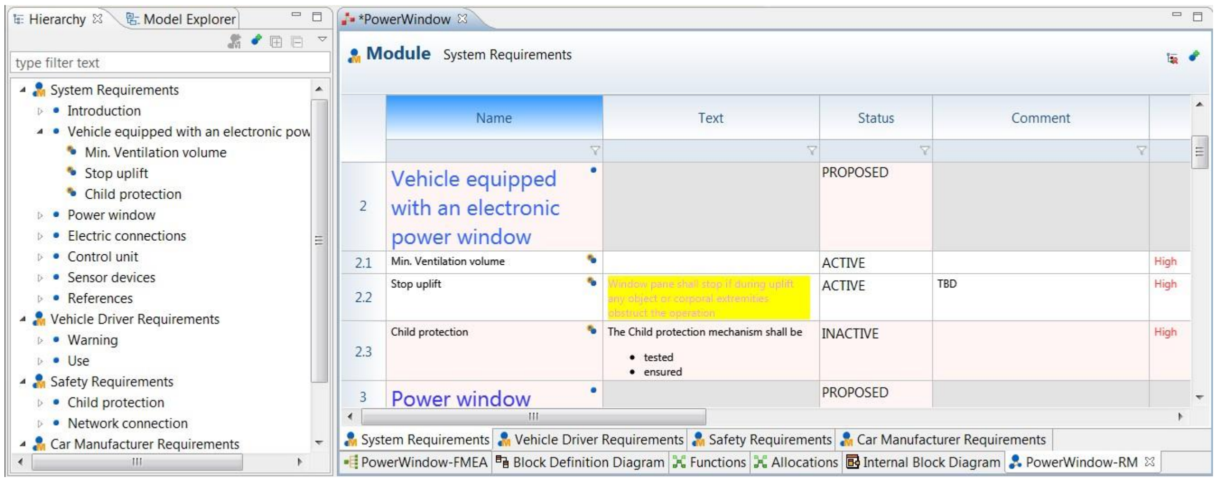
Control Unit [SE4]
- Control the window lifter A
  - Window control not worki [MF12]
  - Window control works out ASILB [MF13]

Sensor Devices [SE5]

PowerWindow-FMEA | Block Definition Diagram | Functions | Allocations | Internal Block Diagram | PowerWindow-RM

## Requirements in RM Module and System Design

- Requirements can be created or imported (ReqIF format) in the RM (Requirements) module.



Hierarchy | Model Explorer

type filter text

- System Requirements
  - Introduction
  - Vehicle equipped with an electronic pow
    - Min. Ventilation volume
    - Stop uplift
    - Child protection
  - Power window
  - Electric connections
  - Control unit
  - Sensor devices
  - References
- Vehicle Driver Requirements
  - Warning
  - Use
- Safety Requirements
  - Child protection
  - Network connection
- Car Manufacturer Requirements

*PowerWindow

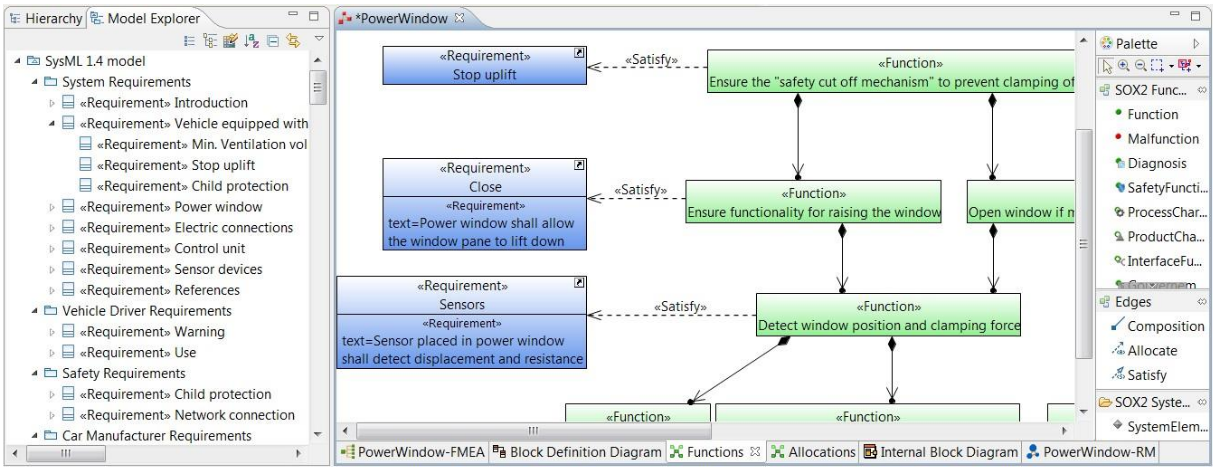
**Module** System Requirements

| | Name | Text | Status | Comment | |
|---|---|---|---|---|---|
| 2 | Vehicle equipped with an electronic power window | | PROPOSED | | |
| 2.1 | Min. Ventilation volume | | ACTIVE | | High |
| 2.2 | Stop uplift | Window pane shall stop if during uplift any object or corporal extremities obstruct the operation | ACTIVE | TBD | High |
| 2.3 | Child protection | The Child protection mechanism shall be<br>• tested<br>• ensured | INACTIVE | | High |
| 3 | Power window | | PROPOSED | | |

System Requirements | Vehicle Driver Requirements | Safety Requirements | Car Manufacturer Requirements

PowerWindow-FMEA | Block Definition Diagram | Functions | Allocations | Internal Block Diagram | PowerWindow-RM

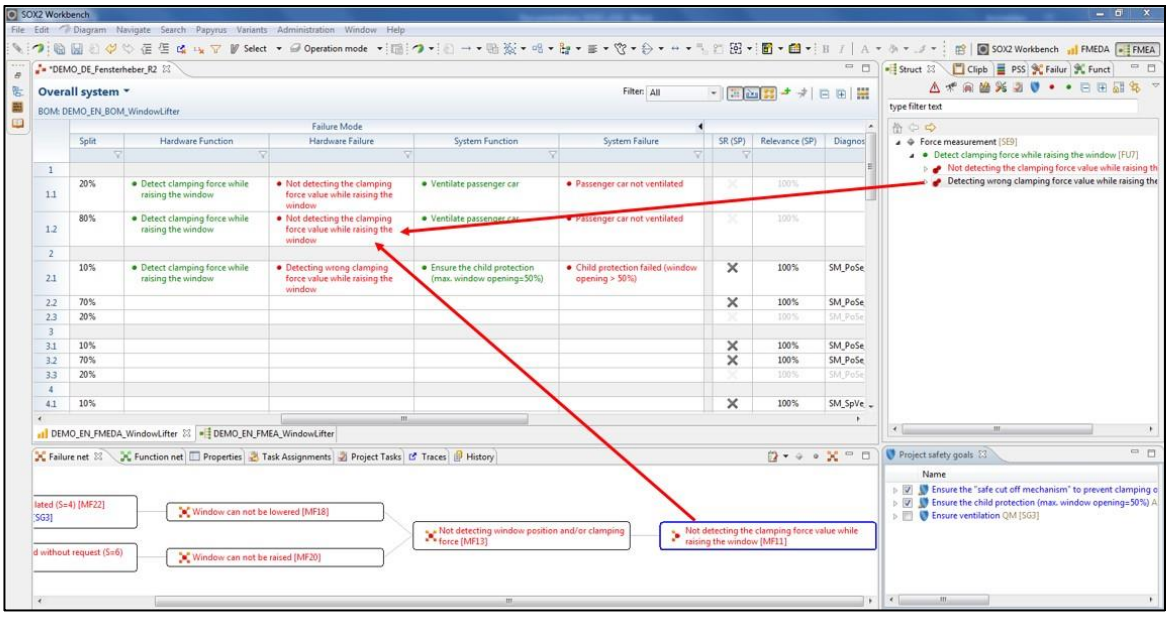
- All added/imported requirements are automatically available in the system design and can be dragged there into diagrams to link them with system design elements (*Satisfy* relationship in SysML).


Hierarchy
Model Explorer
SysML 1.4 model
System Requirements
«Requirement» Introduction
«Requirement» Vehicle equipped with
«Requirement» Min. Ventilation vol
«Requirement» Stop uplift
«Requirement» Child protection
«Requirement» Power window
«Requirement» Electric connections
«Requirement» Control unit
«Requirement» Sensor devices
«Requirement» References
Vehicle Driver Requirements
«Requirement» Warning
«Requirement» Use
Safety Requirements
«Requirement» Child protection
«Requirement» Network connection
Car Manufacturer Requirements
*PowerWindow
«Requirement» Stop uplift
«Satisfy»
«Function» Ensure the "safety cut off mechanism" to prevent clamping of
«Requirement» Close
«Requirement» text=Power window shall allow the window pane to lift down
«Satisfy»
«Function» Ensure functionality for raising the window
Open window if n
«Requirement» Sensors
«Requirement» text=Sensor placed in power window shall detect displacement and resistance
«Satisfy»
«Function» Detect window position and clamping force
«Function»
«Function»
Palette
SOX2 Func...
Function
Malfunction
Diagnosis
SafetyFuncti...
ProcessChar...
ProductCha...
InterfaceFu...
Edges
Composition
Allocate
Satisfy
SOX2 Syste...
SystemElem...
PowerWindow-FMEA
Block Definition Diagram
Functions
Allocations
Internal Block Diagram
PowerWindow-RM

- In turn, requirements created within SysML (in the Model Explorer or Requirements diagram) automatically appear in the RM module. Hence, creating a requirement in SysML requires to select the RM document and module or parent requirement it should be added to. As a consequence, it is not possible to create a requirement in SysML before a RM document has been created.
- Linking requirement with system design elements in other modules (e.g., FMEA editor) can be performed by just dragging a requirement on the desired element. This causes automated creation of a corresponding *Satisfy* relationship in the system design.

## Malfunctions and Functions in FMEDA

- Functions and malfunctions created in SysML 1.4 or FMEA can be used in FMEDA documents as hardware failures and hardware functions.
- Drag and drop malfunctions (e.g., from FMEA "Failure Net" view, FMEA "Structure Content" view, "Model Explorer" view, or "Object List" view) into 'Hardware Failure' column of the corresponding component.
- Functions corresponding to the malfunctions selected will automatically appear in the column 'Hardware Function'.

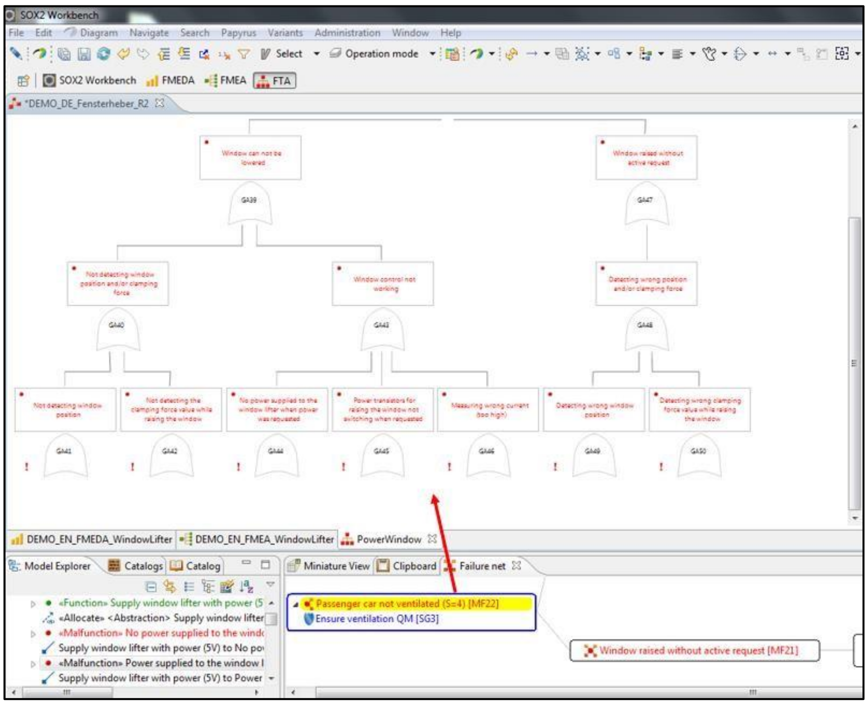

SOX2 Workbench
File Edit Diagram Navigate Search Papyrus Variants Administration Window Help
Select
Operation mode
*DEMO_DE_Fensterheber_R2
Overall system
Filter: All
BOM: DEMO_EN_BOM_WindowLifter
DEMO_EN_FMEDA_WindowLifter
DEMO_EN_FMEA_WindowLifter
Failure net
Function net
Properties
Task Assignments
Project Tasks
Traces
History
lated (S=4) [MF22]
SG3]
d without request (S=6)
Window can not be lowered [MF18]
Window can not be raised [MF20]
Not detecting window position and/or clamping force [MF13]
Not detecting the clamping force value while raising the window [MF11]
Struct
Clipb
PSS
Failur
Funct
SOX2 Workbench
FMEDA
FMEA
type filter text
Force measurement [SE9]
Detect clamping force while raising the window [FU7]
Not detecting the clamping force value while raising th
Detecting wrong clamping force value while raising the
Project safety goals
Name
Ensure the "safe cut off mechanism" to prevent clamping o
Ensure the child protection (max. window opening=50%) A
Ensure ventilation QM [SG3]

| | Failure Mode | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Split | Hardware Function | Hardware Failure | System Function | System Failure | SR (SP) | Relevance (SP) | Diagnos |
| 1 | | | | | | | | |
| 1.1 | 20% | ● Detect clamping force while raising the window | ● Not detecting the clamping force value while raising the window | ● Ventilate passenger car | ● Passenger car not ventilated | ✕ | 100% | |
| 1.2 | 80% | ● Detect clamping force while raising the window | ● Not detecting the clamping force value while raising the window | ● Ventilate passenger car | ● Passenger car not ventilated | ✕ | 100% | |
| 2 | | | | | | | | |
| 2.1 | 10% | ● Detect clamping force while raising the window | ● Detecting wrong clamping force value while raising the window | ● Ensure the child protection (max. window opening=50%) | ● Child protection failed (window opening > 50%) | ✕ | 100% | SM_PoSe |
| 2.2 | 70% | | | | | ✕ | 100% | SM_PoSe |
| 2.3 | 20% | | | | | ✕ | 100% | SM_PoSe |
| 3 | | | | | | | | |
| 3.1 | 10% | | | | | ✕ | 100% | SM_PoSe |
| 3.2 | 70% | | | | | ✕ | 100% | SM_PoSe |
| 3.3 | 20% | | | | | ✕ | 100% | SM_PoSe |
| 4 | | | | | | | | |
| 4.1 | 10% | | | | | ✕ | 100% | SM_SpVe |

- After selecting a system effect, the corresponding system function appears automatically as well.

## Automatically Derive an FTA Structure From an FMEA

- The connections between malfunctions defined in Failure Nets (or the system design, respectively) can be used to automatically derive a corresponding fault tree structure.
- Create an empty FTA document and drag and drop a malfunction into the document (e.g., from FMEA "Failure Net" view, FMEA "Structure Content" view, "Model Explorer" view, or "Object List" view).
- All malfunctions defined as direct or indirect causes in the Failure Net will be automatically added to the fault tree structure.

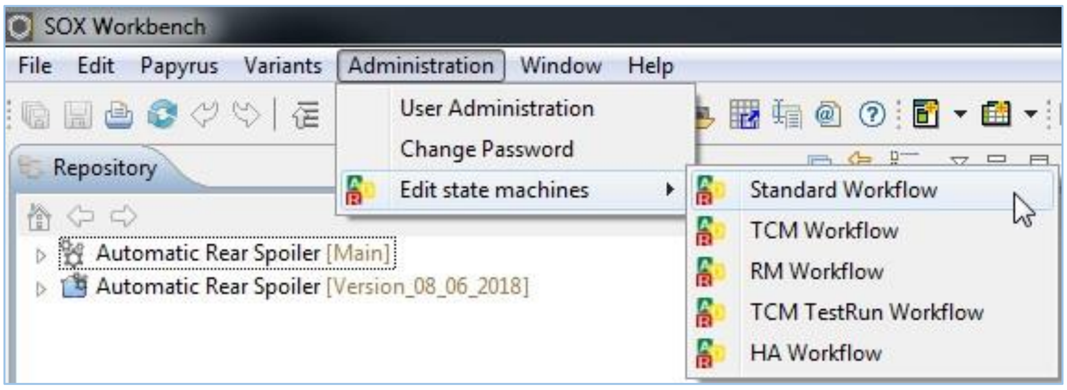

SOX2 Workbench
File Edit Diagram Navigate Search Papyrus Variants Administration Window Help
Select
Operation mode
SOX2 Workbench
FMEDA
FMEA
FTA
*DEMO_DE_Fensterheber_R2
Window can not be lowered
Window raised without active request
GA39
GA47
Not detecting window position and/or clamping force
Window control not working
Detecting wrong position and/or clamping force
GA40
GA43
GA48
Not detecting window position
Not detecting the clamping force value while raising the window
No power supplied to the window lifter when power was requested
Power translators for raising the window not switching when requested
Measuring wrong current (too high)
Detecting wrong window position
Detecting wrong clamping force value while raising the window
GA41
GA42
GA44
GA45
GA46
GA49
GA50
DEMO_EN_FMEDA_WindowLifter
DEMO_EN_FMEA_WindowLifter
PowerWindow
Model Explorer
Catalogs
Catalog
Miniature View
Clipboard
Failure net

«Function» Supply window lifter with power (5
«Allocate» <Abstraction> Supply window lifter
«Malfunction» No power supplied to the windc
Supply window lifter with power (5V) to No po
«Malfunction» Power supplied to the window l
Supply window lifter with power (5V) to Power

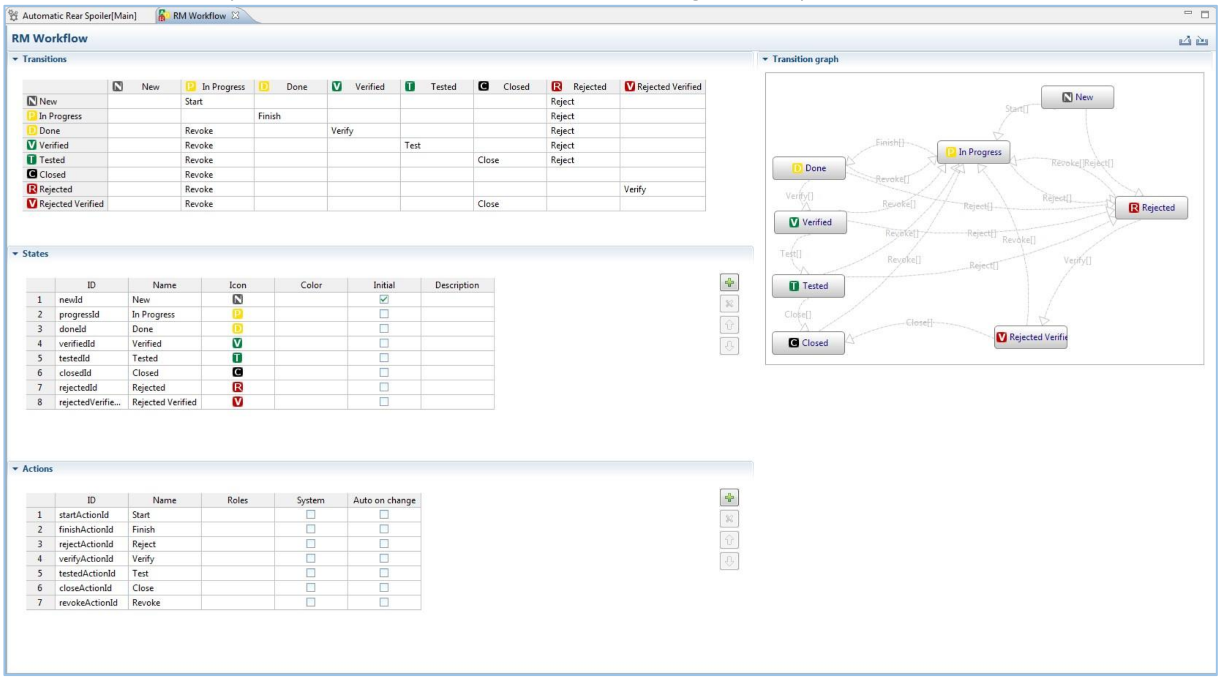
Passenger car not ventilated (S=4) [MF22]
Ensure ventilation QM [SG3]
Window raised without active request [MF21]

## 10. State machine

- Create individual states of SOX elements and define its desired work flow
- A workflow is a set of statuses and transitions that and object moves through during its lifecycle and typically represents processes within your organization. There are individual workflows for the modules Requirements, Test, Hazard, Reliability and FMEDA. These workflows are binded on to the listed modules and can only be used for the according ones. The workflows will be saved in your repository and affects all projects
- SOX provides default standard work flows for each SOX modules which can be edited and adapted ○ *Standard Workflow*: Reliability (BOM) / FMEDA modules ○ *TCM Workflow*: Test Coverage module ○ *RM Workflow*: Requirements module ○ *TCM TestRun*: Part of the Test Coverage module ○ *HA Workflow*: Hazard & Risk Analysis module



- Additionally, defined roles (SOX users) can be assigned to specific states



RM Workflow

Transitions

| | N New | P In Progress | D Done | V Verified | T Tested | C Closed | R Rejected | V Rejected Verified |
|---|---|---|---|---|---|---|---|---|
| N New | | Start | | | | | Reject | |
| P In Progress | | | Finish | | | | Reject | |
| D Done | | Revoke | | Verify | | | Reject | |
| V Verified | | Revoke | | | Test | | Reject | |
| T Tested | | Revoke | | | | Close | Reject | |
| C Closed | | Revoke | | | | | | |
| R Rejected | | Revoke | | | | | | Verify |
| V Rejected Verified | | Revoke | | | | Close | | |

States

| | ID | Name | Icon | Color | Initial | Description |
|---|---|---|---|---|---|---|
| 1 | newId | New | N | | ✓ | |
| 2 | progressId | In Progress | P | | ☐ | |
| 3 | doneId | Done | D | | ☐ | |
| 4 | verifiedId | Verified | V | | ☐ | |
| 5 | testedId | Tested | T | | ☐ | |
| 6 | closedId | Closed | C | | ☐ | |
| 7 | rejectedId | Rejected | R | | ☐ | |
| 8 | rejectedVerifie... | Rejected Verified | V | | ☐ | |

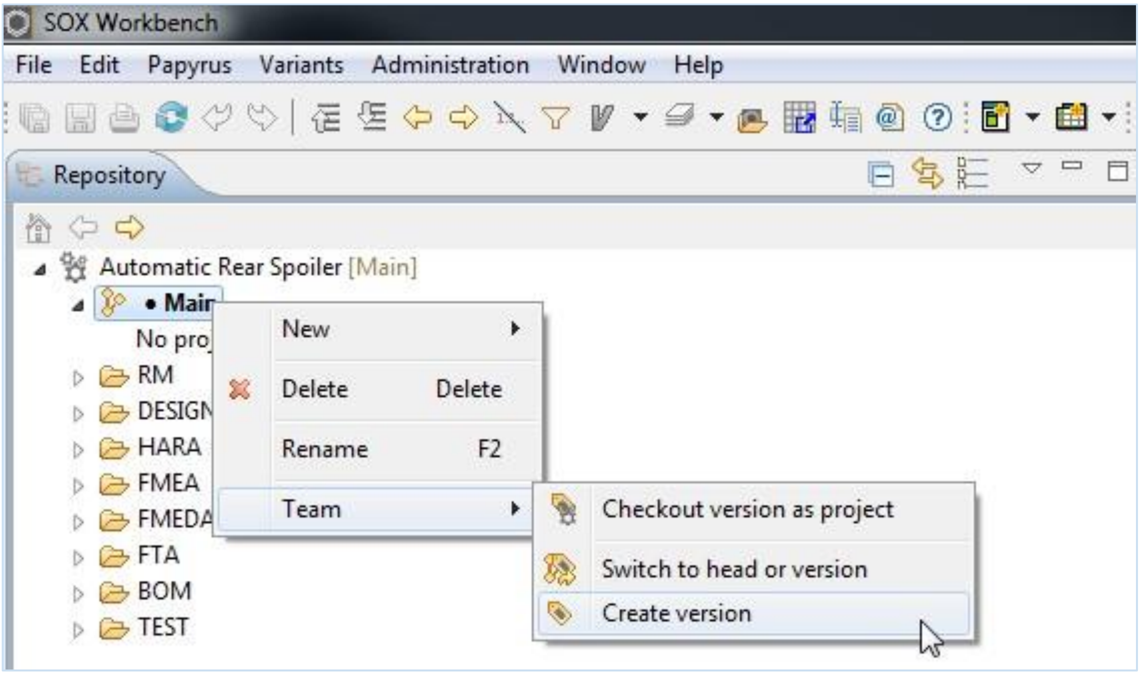
Actions

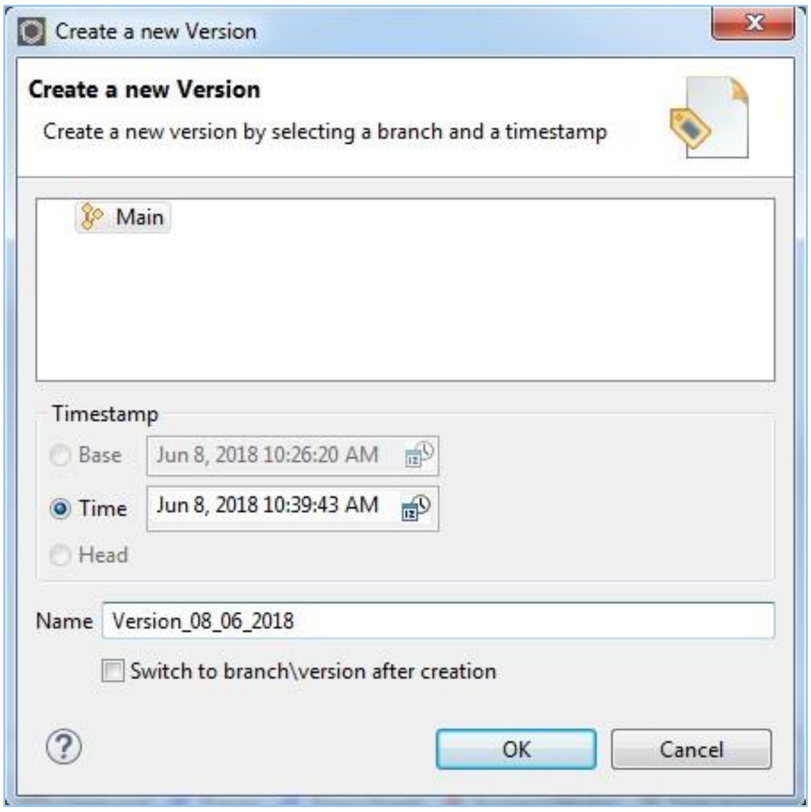
| | ID | Name | Roles | System | Auto on change |
|---|---|---|---|---|---|
| 1 | startActionId | Start | | ☐ | ☐ |
| 2 | finishActionId | Finish | | ☐ | ☐ |
| 3 | rejectActionId | Reject | | ☐ | ☐ |
| 4 | verifyActionId | Verify | | ☐ | ☐ |
| 5 | testedActionId | Test | | ☐ | ☐ |
| 6 | closeActionId | Close | | ☐ | ☐ |
| 7 | revokeActionId | Revoke | | ☐ | ☐ |

- Figure: example for default requirements workflow

## 11. Version management

- Create versions at specific times possible over the entire time of project duration since the repository has been started.
- Versions created within a project cannot be edited. At any time, it is possible to switch back to the main branch.
- Versions can be checked out as separate project file and therefore can be edited and continued to work with.


SOX Workbench
File Edit Papyrus Variants Administration Window Help
Repository
Automatic Rear Spoiler [Main]
• Main
No pro
RM
DESIGN
HARA
FMEA
FMEDA
FTA
BOM
TEST
New
Delete Delete
Rename F2
Team
Checkout version as project
Switch to head or version
Create version


Create a new Version

**Create a new Version**

Create a new version by selecting a branch and a timestamp

Main

Timestamp

Base: Jun 8, 2018 10:26:20 AM

[✓ Time]: Jun 8, 2018 10:39:43 AM

Head

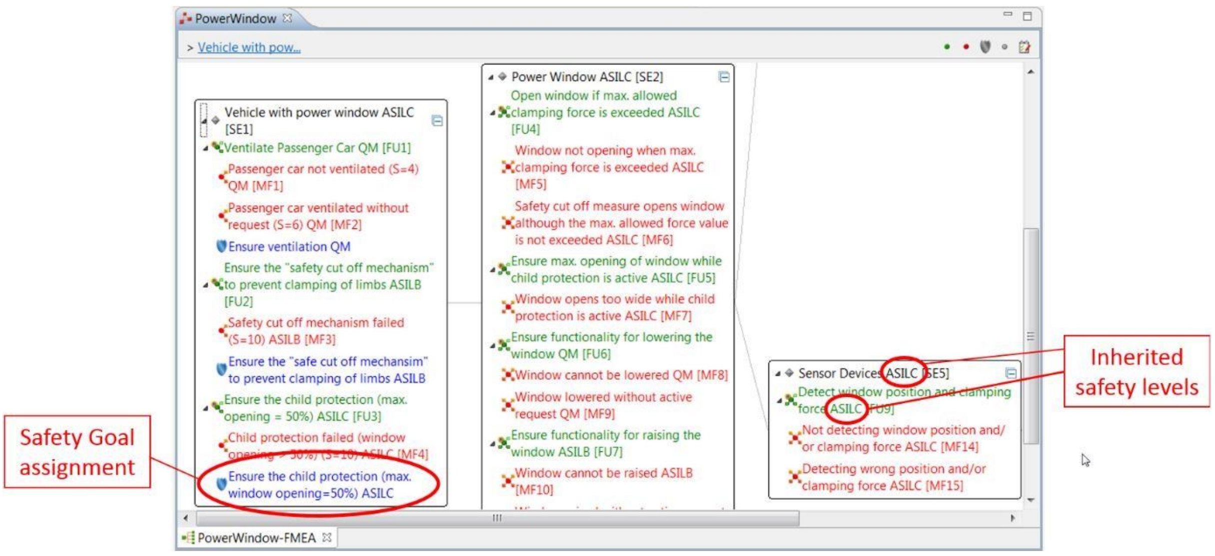
Name: Version_08_06_2018

☐ Switch to branch\version after creation

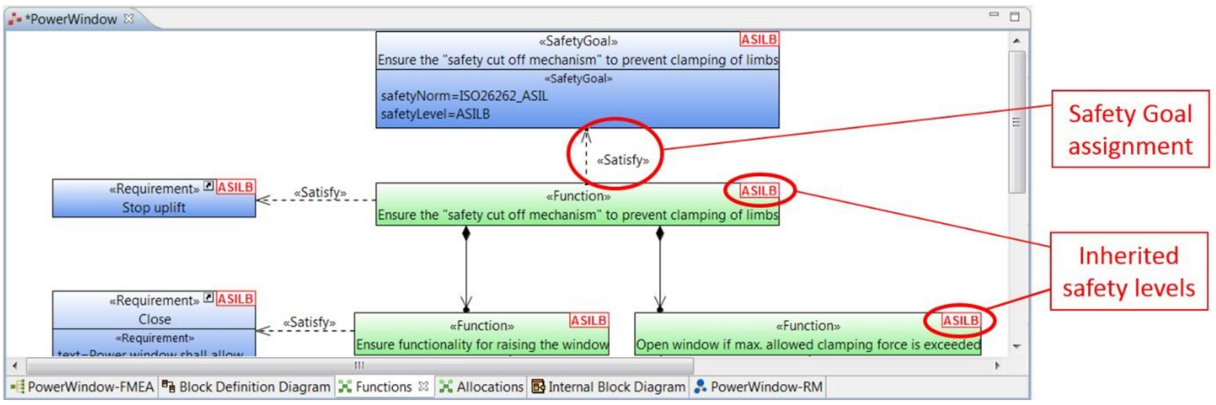
OK Cancel

## 12. Safety Traces

- After assigning safety goals to system design elements, the safety level is automatically inherited to related elements and displayed.
- Create safety goals in “Project Safety Goals” view (right mouse click – new)
- Define customized properties for each safety goal (safety level, norm, safety coverage etc.) • Assign safety goals to functions: ○ In an FMEA document, safety goals are assigned to functions by dragging them onto a function.


PowerWindow
> Vehicle with pow...
Vehicle with power window ASILC [SE1]
Ventilate Passenger Car QM [FU1]
Passenger car not ventilated (S=4) QM [MF1]
Passenger car ventilated without request (S=6) QM [MF2]
Ensure ventilation QM
Ensure the "safety cut off mechanism" to prevent clamping of limbs ASILB [FU2]
Safety cut off mechanism failed (S=10) ASILB [MF3]
Ensure the "safe cut off mechansim" to prevent clamping of limbs ASILB
Ensure the child protection (max. opening = 50%) ASILC [FU3]
Child protection failed (window opening > 50%) (S=10) ASILC [MF4]
Ensure the child protection (max. window opening=50%) ASILC
Power Window ASILC [SE2]
Open window if max. allowed clamping force is exceeded ASILC [FU4]
Window not opening when max. clamping force is exceeded ASILC [MF5]
Safety cut off measure opens window although the max. allowed force value is not exceeded ASILC [MF6]
Ensure max. opening of window while child protection is active ASILC [FU5]
Window opens too wide while child protection is active ASILC [MF7]
Ensure functionality for lowering the window QM [FU6]
Window cannot be lowered QM [MF8]
Window lowered without active request QM [MF9]
Ensure functionality for raising the window ASILB [FU7]
Window cannot be raised ASILB [MF10]
Sensor Devices ASILC [SE5]
Detect window position and clamping force ASILC [FU9]
Not detecting window position and/or clamping force ASILC [MF14]
Detecting wrong position and/or clamping force ASILC [MF15]
PowerWindow-FMEA
Safety Goal assignment
Inherited safety levels

○ In the system design, assignment of safety goals to functions is represented by a SysML *Satisfy* relationship.

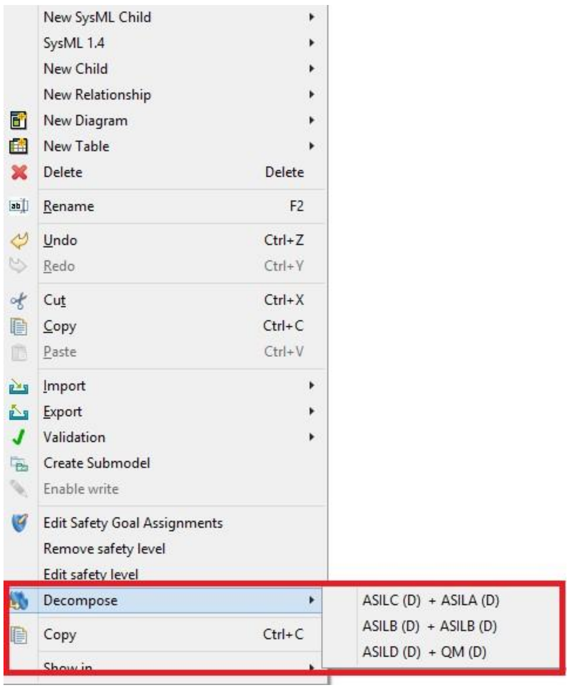


- After assigning the safety goal, the safety levels of all related elements (system elements, functions, malfunctions, requirements) will be automatically calculated and displayed in FMEA documents and SysML diagrams. If an element is affected by multiple safety levels, the more critical one is effective.

## 13. ASIL Decomposition

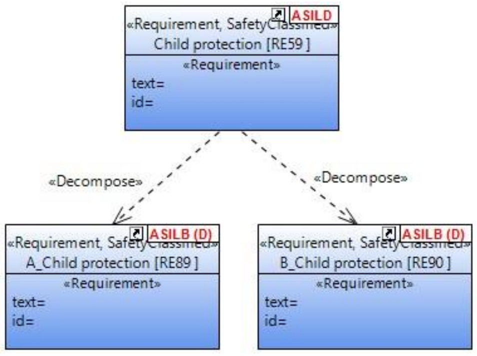
- SOX supports ASIL decomposition according to ISO26262. An ASIL decomposition can be performed on safety goals and requirements with safety classification.

- Safety classifications can be added in the model explorer, in the RM Hierarchy view, PSS view and in the according diagram.
  Right-click on the requirement and choose **Add safety level**.
- Performing an ASIL decomposition:
  Right-click on a safety goal/requirement and choose **Decompose**.

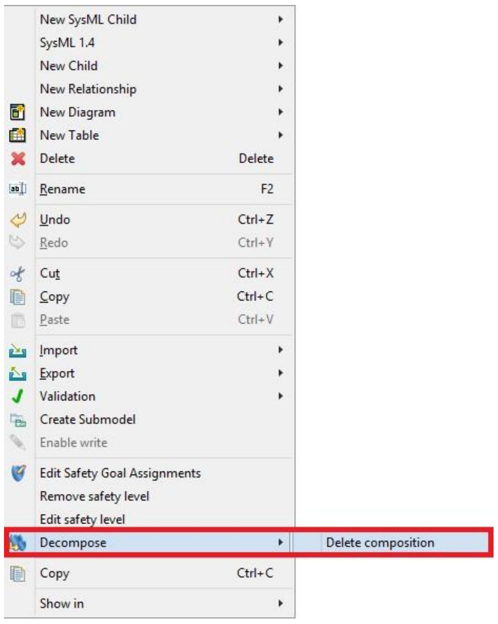


- Deleting and editing the safety classification can also be done in the context menu:

- Below the safety goal/requirement there will be new safety goals/requirements now. Their names start with "A_" and "B_".
- This is how the requirement looks in the diagram:

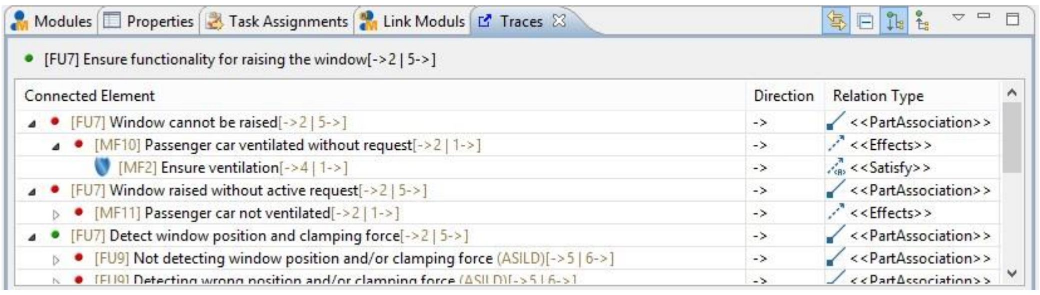

«Requirement, SafetyClassified»
ASILD
Child protection [RE59 ]
«Requirement»
text=
id=
«Decompose»
«Decompose»
«Requirement, SafetyClassified»
ASILB (D)
A_Child protection [RE89 ]
«Requirement»
text=
id=
«Requirement, SafetyClassified»
ASILB (D)
B_Child protection [RE90 ]
«Requirement»
text=
id=

- Deleting an ASIL decomposition:



## 14. Traces view

- The Traces view shows how an object is connected to other objects within the same project.



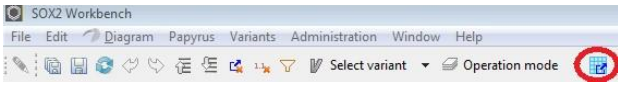
● [FU7] Ensure functionality for raising the window[->2 | 5->]

| Connected Element | Direction | Relation Type |
|---|---|---|
| ● [FU7] Window cannot be raised[->2 \| 5->] | -> | <<PartAssociation>> |
| ● [MF10] Passenger car ventilated without request[->2 \| 1->] | -> | <<Effects>> |
| [MF2] Ensure ventilation[->4 \| 1->] | -> | <<Satisfy>> |
| ● [FU7] Window raised without active request[->2 \| 5->] | -> | <<PartAssociation>> |
| ● [MF11] Passenger car not ventilated[->2 \| 1->] | -> | <<Effects>> |
| ● [FU7] Detect window position and clamping force[->2 \| 5->] | -> | <<PartAssociation>> |
| ● [FU9] Not detecting window position and/or clamping force (ASILD)[->5 \| 6->] | -> | <<PartAssociation>> |
| ● [FU9] Detecting wrong position and/or clamping force (ASILD)[->5 \| 6->] | -> | <<PartAssociation>> |

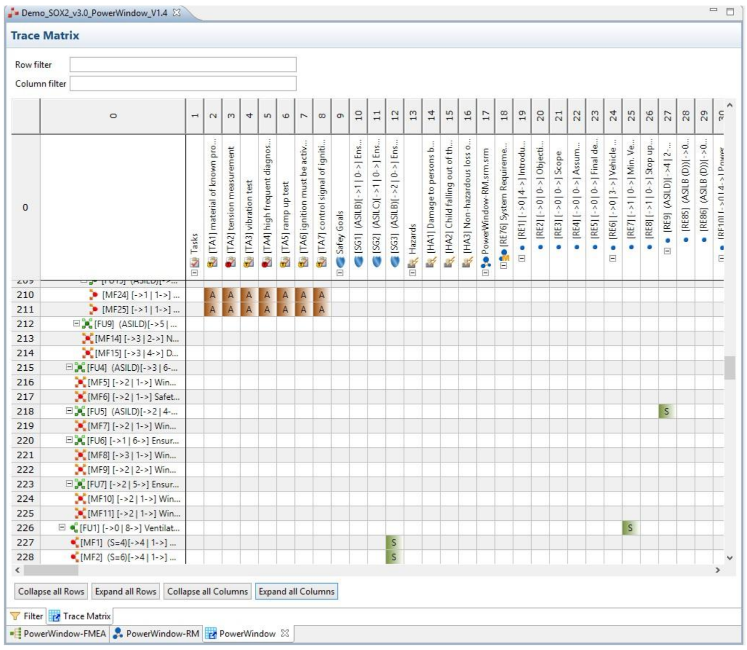
- Select an object in any view and the Traces view will show the traces for this object.
- The buttons "Show incoming references" and "Show outgoing references" filter for different traces:

## 15. The Trace Matrix

- The Trace Matrix shows all the connections between objects in a project.
- It is located in the menu bar:



- Here is an example:



Trace Matrix

Row filter

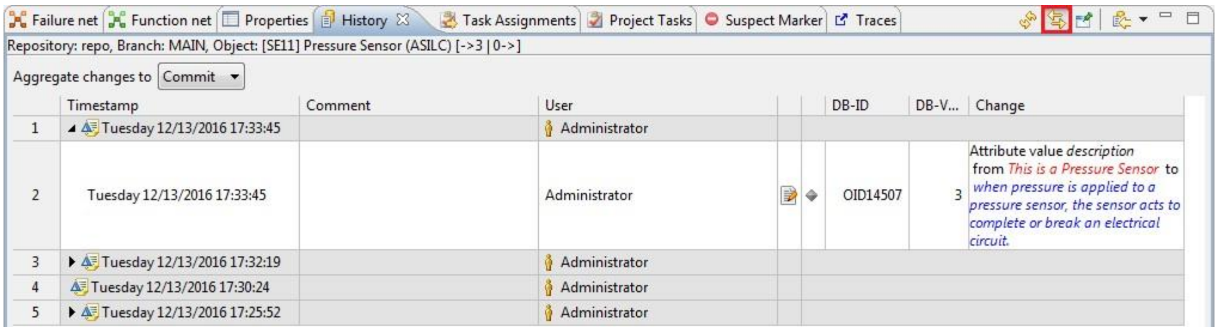
Column filter

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | Tasks | [TA1] material of known pro... | [TA2] tension measurement | [TA3] vibration test | [TA4] high frequent diagnos... | [TA5] ramp up test | [TA6] ignition must be activ... | [TA7] control signal of igniti... | Safey Goals | [SG1] (ASILB)[->1 \| 0->] Ens... | [SG2] (ASILC)[->1 \| 0->] Ens... | [SG3] (ASILB)[->2 \| 0->] Ens... | Hazards | [HA1] Damage to persons b... | [HA2] Child falling out of th... | [HA3] Non-hazardous loss o... | PowerWindow-RM.srm.srm | [RE76] System Requireme... | [RE1] [->0 \| 4->] Introdu... | [RE2] [->0 \| 0->] Objecti... | [RE3] [->0 \| 0->] Scope | [RE4] [->0 \| 0->] Assum... | [RE5] [->0 \| 0->] Final de... | [RE6] [->0 \| 3->] Vehicle ... | [RE7] [->1 \| 0->] Min. Ve... | [RE8] [->1 \| 0->] Stop up... | [RE9] (ASILD)[->4 \| 2-... | [RE85] (ASILB (D))[->0... | [RE86] (ASILB (D))[->0... | [RE101] [->0 \| 4->] Power |
| 210 | [MF24] [->1 \| 1->] ... | | A | A | A | A | A | A | A | | | | | | | | | | | | | | | | | | | | | | |
| 211 | [MF25] [->1 \| 1->] ... | | A | A | A | A | A | A | A | | | | | | | | | | | | | | | | | | | | | | |
| 212 | [FU9] (ASILD)[->5 \| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 213 | [MF14] [->3 \| 2->] N... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 214 | [MF15] [->3 \| 4->] D... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 215 | [FU4] (ASILD)[->3 \| 6-... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 216 | [MF5] [->2 \| 1->] Win... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 217 | [MF6] [->2 \| 1->] Safet... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 218 | [FU5] (ASILD)[->2 \| 4-... | | | | | | | | | | | | | | | | | | | | | | | | | | | S | | | |
| 219 | [MF7] [->2 \| 1->] Win... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 220 | [FU6] [->1 \| 6->] Ensur... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 221 | [MF8] [->3 \| 1->] Win... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 222 | [MF9] [->2 \| 2->] Win... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 223 | [FU7] [->2 \| 5->] Ensur... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 224 | [MF10] [->2 \| 1->] Win... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 225 | [MF11] [->2 \| 1->] Win... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 226 | [FU1] [->0 \| 8->] Ventilat... | | | | | | | | | | | | | | | | | | | | | | | | | S | | | | | |
| 227 | [MF1] (S=4)[->4 \| 1->] ... | | | | | | | | | | | S | | | | | | | | | | | | | | | | | | | |
| 228 | [MF2] (S=6)[->4 \| 1->] ... | | | | | | | | | | | S | | | | | | | | | | | | | | | | | | | |

- The colors in the table fade to white either to the left or to the right. This indicates the direction of the relationship.
- Multiple filters help to find what is needed.

## 16. History View

Every change to a document is tracked and stored in the database. Users can view the history of changes performed on a document in the History view.



Repository: repo, Branch: MAIN, Object: [SE11] Pressure Sensor (ASILC) [->3 | 0->]

Aggregate changes to: Commit

| | Timestamp | Comment | User | | | DB-ID | DB-V... | Change |
|---|---|---|---|---|---|---|---|---|
| 1 | Tuesday 12/13/2016 17:33:45 | | Administrator | | | | | |
| 2 | Tuesday 12/13/2016 17:33:45 | | Administrator | | | OID14507 | 3 | Attribute value *description* from *This is a Pressure Sensor* to *when pressure is applied to a pressure sensor, the sensor acts to complete or break an electrical circuit.* |
| 3 | Tuesday 12/13/2016 17:32:19 | | Administrator | | | | | |
| 4 | Tuesday 12/13/2016 17:30:24 | | Administrator | | | | | |
| 5 | Tuesday 12/13/2016 17:25:52 | | Administrator | | | | | |

- The “Link with Editor” option (highlighted in red in the screenshot above) in the History view must be activated to display the history of the currently selected object.

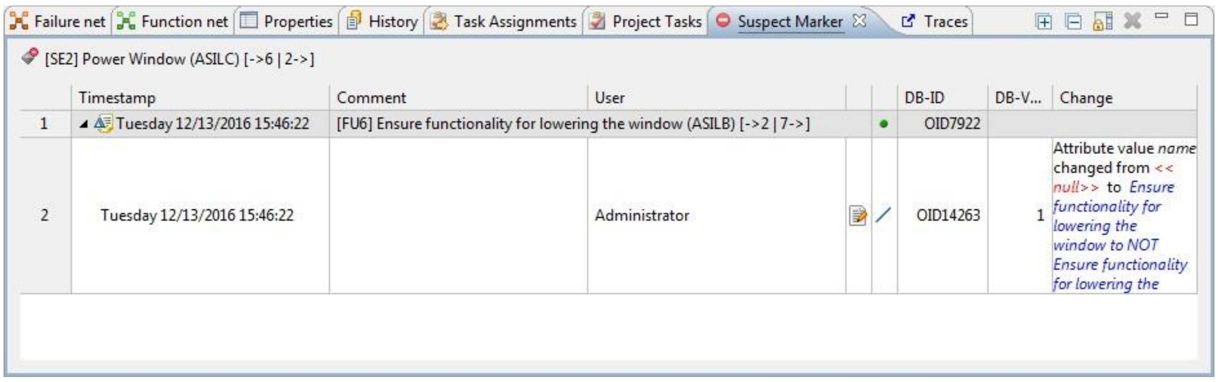
- All changes can be aggregated according to the following options: commit, hour, day, week or month. For instance, when selecting “hour”, all changes within an hour are summarized in one line, while “commit” displays every single change.

- Expand a single entry to see its details (if any) by clicking on the small rectangle on its left.
- In the current version, only the history of objects that have a SysML representation (e.g., requirements, function, system elements) is displayed and the proper display of change information is not fully working.

## 17. Suspect Marker View

Changes to an object (e.g., requirement) can have effects on other linked objects (e.g., linked Functions). Hence, after a change was performed, all linked objects might need to be reviewed and potentially to be adapted. For instance, after a requirement was changed, all linked functions might need to be adapted. For this, all linked objects are automatically marked as “suspect” which is indicated by a red circle on the object.

The Suspect Marker view displays the causes of “suspect” markers for a selected object. Once a suspect object has been reviewed, the marker can be deleted by the responsible user in the Suspect Marker view. It is also possible to select a whole project and to inspect all suspect objects in the project.

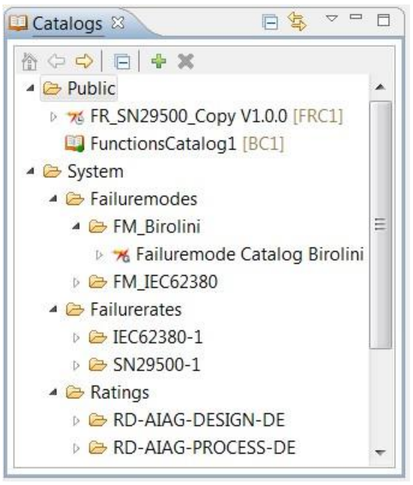


[SE2] Power Window (ASILC) [->6 | 2->]

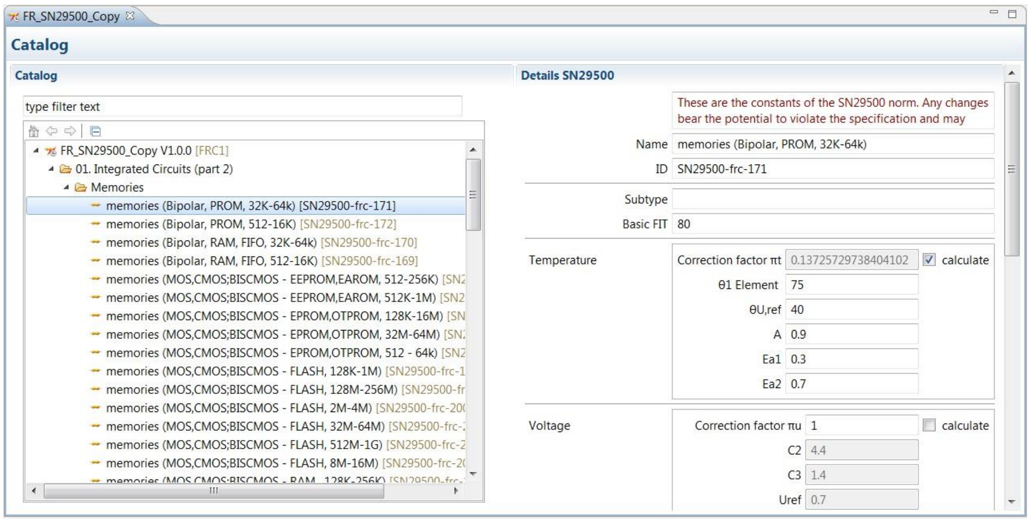
| | Timestamp | Comment | User | | | DB-ID | DB-V... | Change |
|---|---|---|---|---|---|---|---|---|
| 1 | Tuesday 12/13/2016 15:46:22 | [FU6] Ensure functionality for lowering the window (ASILB) [->2 \| 7->] | | | ● | OID7922 | | |
| 2 | Tuesday 12/13/2016 15:46:22 | | Administrator | | | OID14263 | 1 | Attribute value *name* changed from *<<null>>* to *Ensure functionality for lowering the window to NOT Ensure functionality for lowering the* |

# 18.Server-based Catalogs

- SOX version 3.0 supports managing and storing catalogs on a SOX server to share them across a whole organization or department.
- Catalogs are used and managed using the “Catalogs” view which displays the catalogs available in the repository the user is connected to (remote repository or local repository; server-based sharing of catalogs is only possible when connected to a remote repository).



- The catalogs are structured into two folders:
  - *System*: Standard catalogs for norms and standards provided by SOX. The content of this folder cannot be modified except the employee catalog.
  - *Public*: Custom catalogs. This includes custom versions of the standard catalogs as well as newly created catalogs for common reusable objects (e.g., system elements, functions, malfunctions, machines) to be shared with other projects.
- To create a custom version of a standard catalog, copy a standard catalog (right-click on a standard catalog and select *copy*) and paste it into the public folder or a subfolder of it (rightclick on the folder and *paste*). A copy of the catalog now appears at the chosen location that can be edited using the “Catalog Editor” (see below).
- To create a custom catalog for reusable elements, right-click on the public folder (or any subfolder), select *New* and the type of catalog (e.g., Function Catalog).
- Failure mode, failure rate, inspection, machine and rating catalogs can be edited by doubleclicking them which opens the “Catalog Editor”. Its left part shows the catalog items, its right part shows the details of the item selected on the left side. It depends on the type of item which details are displayed and which of them can be edited. (The checkbox calculate in standard catalogs defines if the associated value is calculated based on its parameters listed below or if the fixed value in the field next to checkbox is used instead).



FR_SN29500_Copy

**Catalog**

Catalog

type filter text

- FR_SN29500_Copy V1.0.0 [FRC1]
  - 01. Integrated Circuits (part 2)
    - Memories
      - memories (Bipolar, PROM, 32K-64k) [SN29500-frc-171]
      - memories (Bipolar, PROM, 512-16K) [SN29500-frc-172]
      - memories (Bipolar, RAM, FIFO, 32K-64k) [SN29500-frc-170]
      - memories (Bipolar, RAM, FIFO, 512-16K) [SN29500-frc-169]
      - memories (MOS,CMOS;BISCMOS - EEPROM,EAROM, 512-256K) [SN2
      - memories (MOS,CMOS;BISCMOS - EEPROM,EAROM, 512K-1M) [SN2
      - memories (MOS,CMOS;BISCMOS - EPROM,OTPROM, 128K-16M) [SN
      - memories (MOS,CMOS;BISCMOS - EPROM,OTPROM, 32M-64M) [SN
      - memories (MOS,CMOS;BISCMOS - EPROM,OTPROM, 512 - 64k) [SN2
      - memories (MOS,CMOS;BISCMOS - FLASH, 128K-1M) [SN29500-frc-1
      - memories (MOS,CMOS;BISCMOS - FLASH, 128M-256M) [SN29500-fr
      - memories (MOS,CMOS;BISCMOS - FLASH, 2M-4M) [SN29500-frc-20
      - memories (MOS,CMOS;BISCMOS - FLASH, 32M-64M) [SN29500-frc-
      - memories (MOS,CMOS;BISCMOS - FLASH, 512M-1G) [SN29500-frc-2
      - memories (MOS,CMOS;BISCMOS - FLASH, 8M-16M) [SN29500-frc-2

Details SN29500

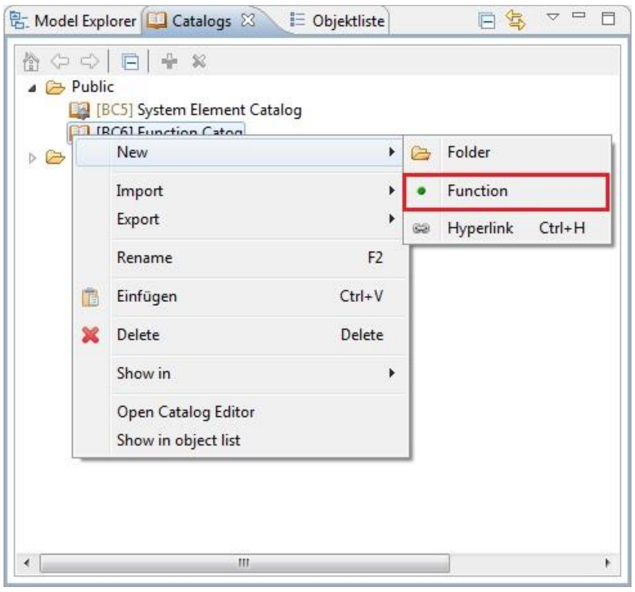
These are the constants of the SN29500 norm. Any changes bear the potential to violate the specification and may

Name: memories (Bipolar, PROM, 32K-64k)
ID: SN29500-frc-171
Subtype:
Basic FIT: 80

Temperature
Correction factor πt: 0.13725729738404102 [✓ calculate]
θ1 Element: 75
θU,ref: 40
A: 0.9
Ea1: 0.3
Ea2: 0.7

Voltage
Correction factor πu: 1 [ ] calculate
C2: 4.4
C3: 1.4
Uref: 0.7

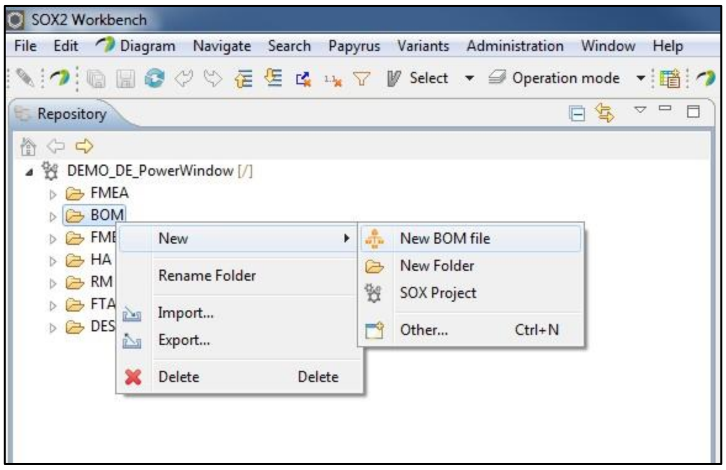
- In case of a catalog for reusable objects (system elements, functions, malfunctions), the content can be added via the context menu directly at the catalog view (right-click on catalog or parent item -> *New*).



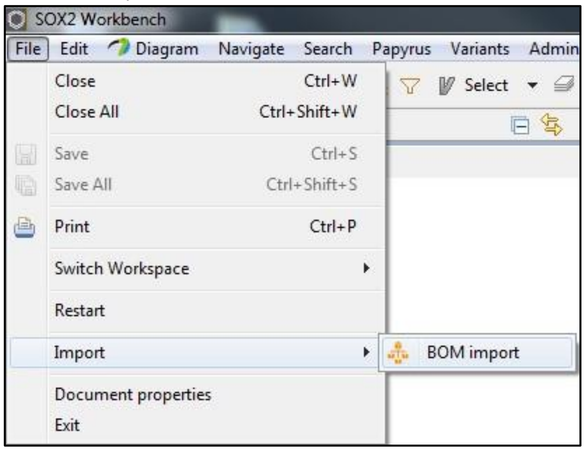
- In catalogs for reusable objects, it is also possible (depending on the item type) to open a more specific property dialog to edit an item via its context menu (right-click on the item -> *Properties*).

## 19. Import of BOM files and assignment of modules to system elements

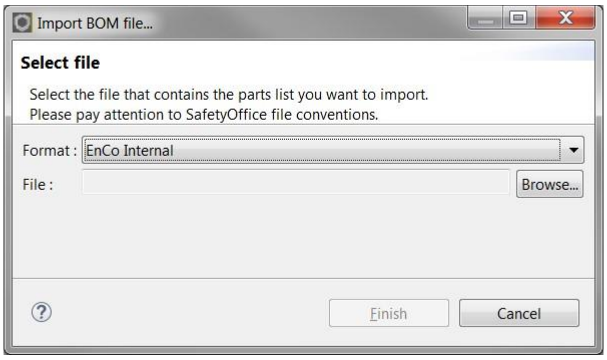
- Import BOM (bill of materials) by using an Excel sheet that corresponds to one of the provided templates (see Excel file templates in delivery package).
- Create a new BOM file (right-click on BOM folder –> select *New BOM File*).



- Choose a file name and select predefined failure mode (Birolini, IEC 62380) and component catalogs (SN 29500, IEC 62380).
- Properties of catalogs can be changed and adapted during the further working process.
- Now select ‘File’, ‘Import’ and ‘BOM Import’ (make sure the created BOM file is still the active document).



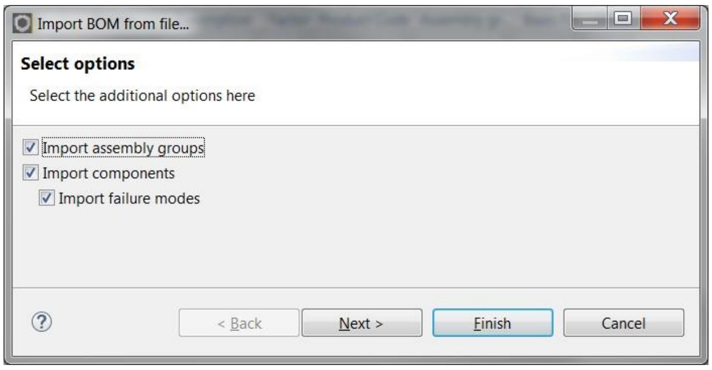
- Select EnCo Internal format and your BOM Excel file.



Select the file that contains the parts list you want to import.
Please pay attention to SafetyOffice file conventions.

Format : EnCo Internal
File :

Optionally, you can select which parts to omit from the import. At this point you could press *Finish* already but selecting *Next* provides you with further options described in the following.

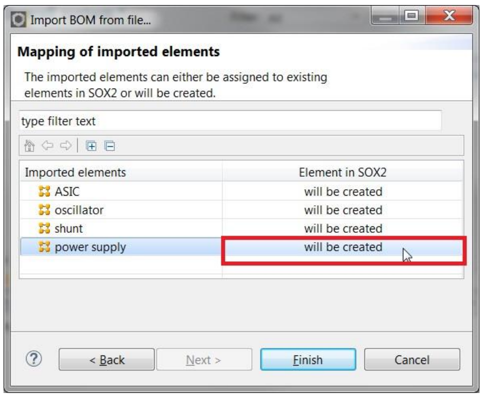

Import BOM from file...

**Select options**

Select the additional options here

- [✓] Import assembly groups
- [✓] Import components
  - [✓] Import failure modes

After clicking *Next* you can map modules contained in your BOM file to already existing modules (system elements) in your project. If you map a module to an existing one, they will be handled as the same element (i.e., merged). If no mapping is specified a new module will be created (indicated by the text "will be created"). A mapping is defined as follows: ○ Click on the right column for the element for which a mapping should be defined.


Import BOM from file...

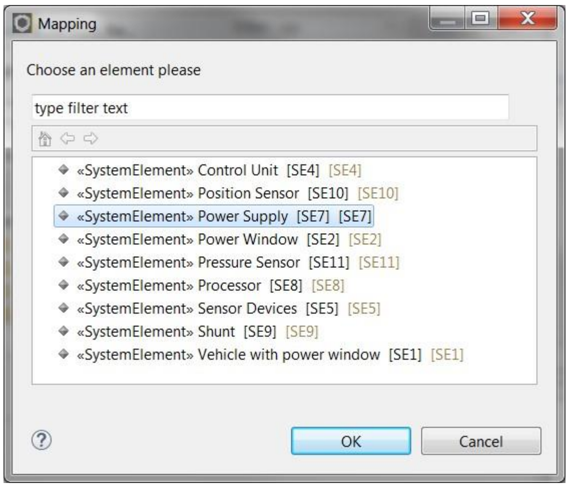
**Mapping of imported elements**

The imported elements can either be assigned to existing elements in SOX2 or will be created.

type filter text

| Imported elements | Element in SOX2 |
|---|---|
| ASIC | will be created |
| oscillator | will be created |
| shunt | will be created |
| power supply | will be created |

- In the upcoming dialog, select an existing element in your project where the imported element should be mapped to.

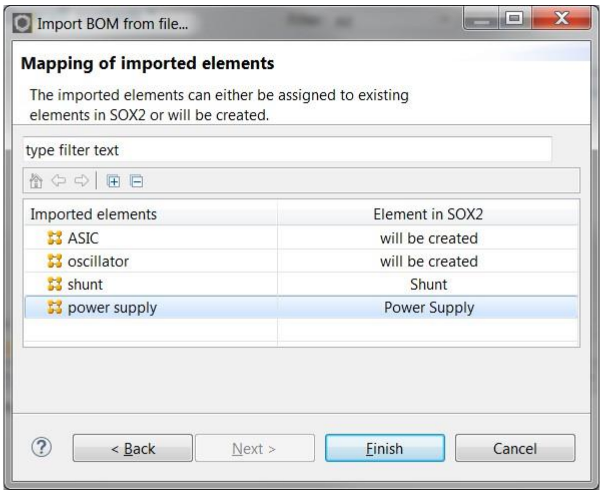

Mapping

Choose an element please

type filter text

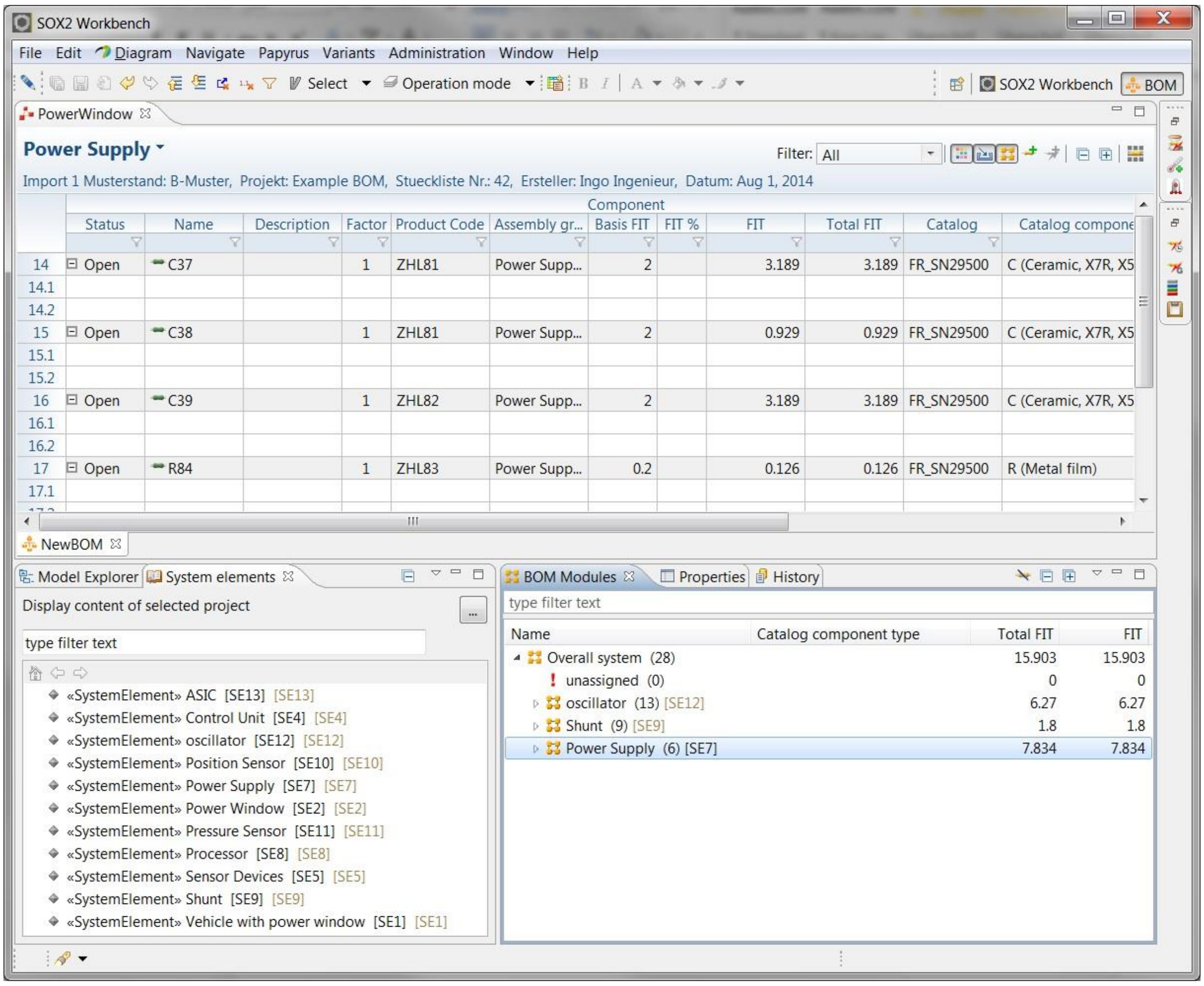
- «SystemElement» Control Unit [SE4] [SE4]
- «SystemElement» Position Sensor [SE10] [SE10]
- «SystemElement» Power Supply [SE7] [SE7]
- «SystemElement» Power Window [SE2] [SE2]
- «SystemElement» Pressure Sensor [SE11] [SE11]
- «SystemElement» Processor [SE8] [SE8]
- «SystemElement» Sensor Devices [SE5] [SE5]
- «SystemElement» Shunt [SE9] [SE9]
- «SystemElement» Vehicle with power window [SE1] [SE1]

- Repeat these steps until all desired mappings are defined and click *Finish*.


Import BOM from file...
Mapping of imported elements
The imported elements can either be assigned to existing elements in SOX2 or will be created.
type filter text
< Back
Next >
Finish
Cancel

| Imported elements | Element in SOX2 |
|---|---|
| ASIC | will be created |
| oscillator | will be created |
| shunt | Shunt |
| power supply | Power Supply |

- If mappings were defined as above, those imported modules will not be newly created but the existing module/system element is used instead. For any other imported modules, a new module (system element) is created.
- In addition, the import automatically merges imported components with existing ones in the BOM file (if applicable) via their names. The merge result is indicated by the color of the component icon in the resulting BOM file:
    - Green: The component has been added to the BOM as result of the import.
    - Yellow: The component already existed in the BOM and has been merged with the corresponding imported one.
    - Red: The component already existed in the BOM but is not present in the imported BOM file.


SOX2 Workbench
File Edit Diagram Navigate Papyrus Variants Administration Window Help
Select Operation mode
SOX2 Workbench BOM
PowerWindow
NewBOM

**Power Supply**

Filter: All

Import 1 Musterstand: B-Muster, Projekt: Example BOM, Stueckliste Nr.: 42, Ersteller: Ingo Ingenieur, Datum: Aug 1, 2014

| | Status | Name | Description | Factor | Product Code | Assembly gr... | Component Basis FIT | FIT % | FIT | Total FIT | Catalog | Catalog compone |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | Open | C37 | | 1 | ZHL81 | Power Supp... | 2 | | 3.189 | 3.189 | FR_SN29500 | C (Ceramic, X7R, X5 |
| 14.1 | | | | | | | | | | | | |
| 14.2 | | | | | | | | | | | | |
| 15 | Open | C38 | | 1 | ZHL81 | Power Supp... | 2 | | 0.929 | 0.929 | FR_SN29500 | C (Ceramic, X7R, X5 |
| 15.1 | | | | | | | | | | | | |
| 15.2 | | | | | | | | | | | | |
| 16 | Open | C39 | | 1 | ZHL82 | Power Supp... | 2 | | 3.189 | 3.189 | FR_SN29500 | C (Ceramic, X7R, X5 |
| 16.1 | | | | | | | | | | | | |
| 16.2 | | | | | | | | | | | | |
| 17 | Open | R84 | | 1 | ZHL83 | Power Supp... | 0.2 | | 0.126 | 0.126 | FR_SN29500 | R (Metal film) |
| 17.1 | | | | | | | | | | | | |

Model Explorer | System elements

Display content of selected project

type filter text

- «SystemElement» ASIC [SE13] [SE13]
- «SystemElement» Control Unit [SE4] [SE4]
- «SystemElement» oscillator [SE12] [SE12]
- «SystemElement» Position Sensor [SE10] [SE10]
- «SystemElement» Power Supply [SE7] [SE7]
- «SystemElement» Power Window [SE2] [SE2]
- «SystemElement» Pressure Sensor [SE11] [SE11]
- «SystemElement» Processor [SE8] [SE8]
- «SystemElement» Sensor Devices [SE5] [SE5]
- «SystemElement» Shunt [SE9] [SE9]
- «SystemElement» Vehicle with power window [SE1] [SE1]

BOM Modules | Properties | History

type filter text

| Name | Catalog component type | Total FIT | FIT |
|---|---|---|---|
| Overall system (28) | | 15.903 | 15.903 |
| ! unassigned (0) | | 0 | 0 |
| oscillator (13) [SE12] | | 6.27 | 6.27 |
| Shunt (9) [SE9] | | 1.8 | 1.8 |
| Power Supply (6) [SE7] | | 7.834 | 7.834 |

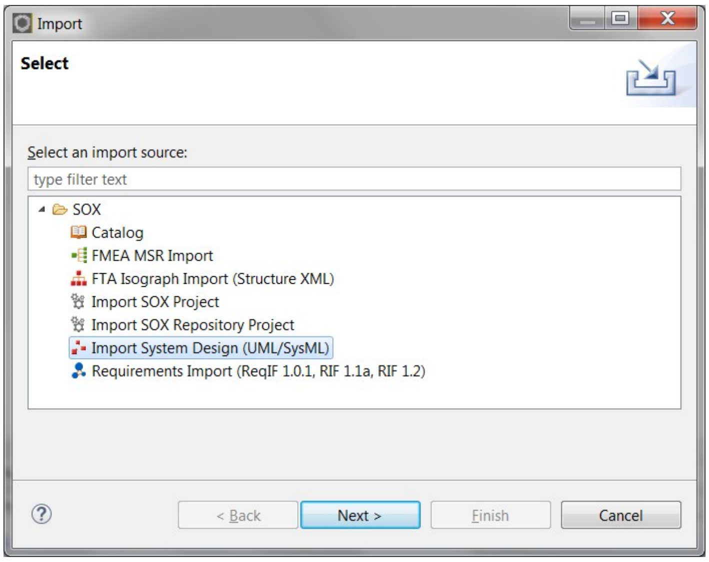
# 20. Import of UML/SysML from Enterprise Architect (Experimental)

It is possible to import a system design (UML/SysML) created with the tool “Enterprise Architect” (EA) into an existing project. Only the model elements (i.e., the data visible in the Model Explorer) can be imported – it is not possible to import the visual diagram information. During the import it is supported to apply stereotypes (System Element, Function, Malfunction) to imported model elements so that they can be directly used for safety analysis in SOX, e.g., by automatically deriving an FMEA structure.

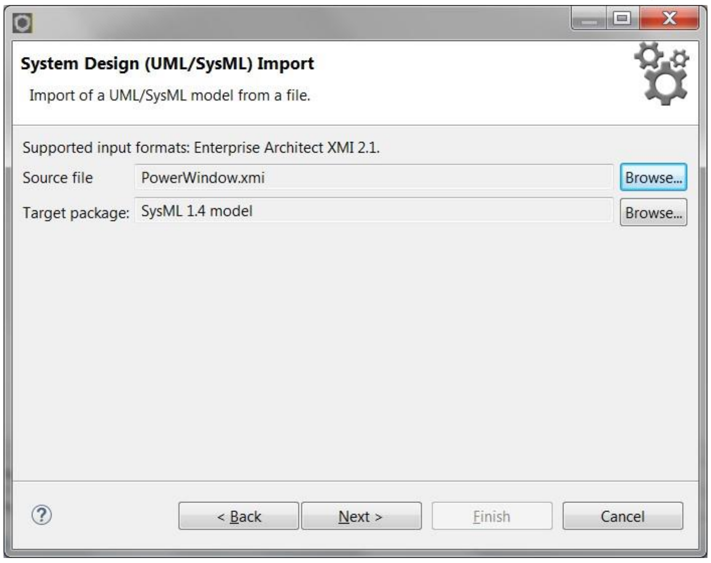
The EA import feature is currently in experimental stage and cannot always import the complete EA models. This is because XMI exported from EA does not fully comply with the UML2/SysML standard. Such incompatible model constructs cannot always be resolved during import and are then ignored. However, the basic structure of a UML/SysML model, such as blocks and allocations, are usually imported. Custom profiles and libraries are not part of the import yet.

Importing a system design from EA into an existing project is performed as follows:

- Right-click on an existing project in the Repository view and choose *Import…*. The selected project must already be open (loaded) in you workspace. To import into a new project, just create a new project as target for the import.
- Select *Import System Design (UML/SysML)*.


Import
Select
Select an import source:
type filter text
SOX
Catalog
FMEA MSR Import
FTA Isograph Import (Structure XML)
Import SOX Project
Import SOX Repository Project
Import System Design (UML/SysML)
Requirements Import (ReqIF 1.0.1, RIF 1.1a, RIF 1.2)
< Back
Next >
Finish
Cancel

- Select the source file and a target package. The source file must be an XMI 2.1 file exported from EA. (It is recommended to export without diagram information and EA extensions for faster processing in SOX but this does not influence the import result.) The target package is a package from your existing model (e.g., the root model *SysML 1.4 model*). The imported model will then be added as a sub-model into this target package.

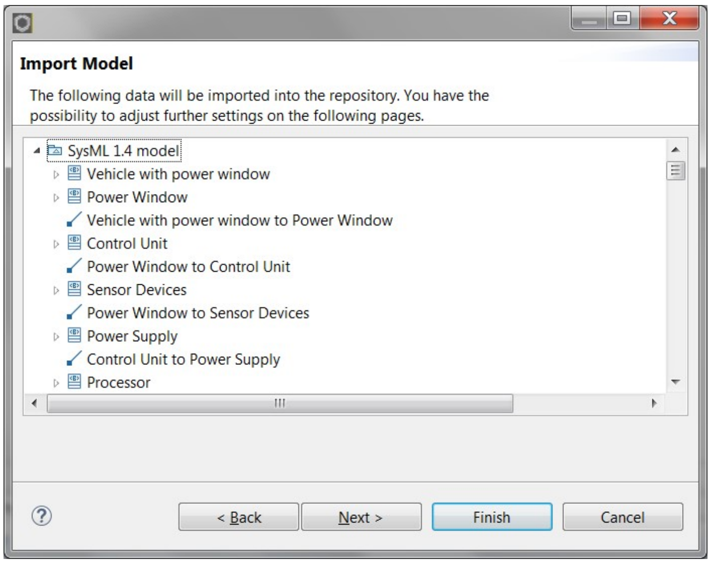

System Design (UML/SysML) Import
Import of a UML/SysML model from a file.
Browse...
Browse...
< Back
Next >
Finish
Cancel

Supported input formats: Enterprise Architect XMI 2.1.

Source file: PowerWindow.xmi

Target package: SysML 1.4 model

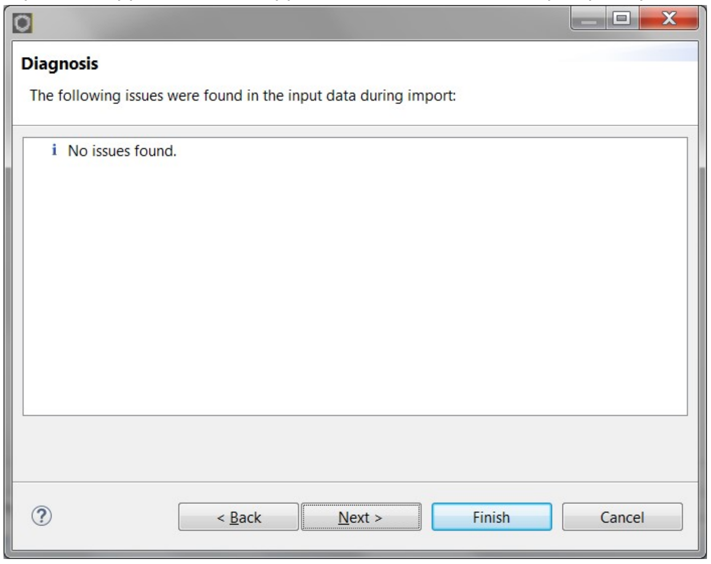
- After clicking next and a short waiting time for processing the import, all recognized model elements are displayed.


**Import Model**

The following data will be imported into the repository. You have the possibility to adjust further settings on the following pages.

- SysML 1.4 model
  - Vehicle with power window
  - Power Window
  - Vehicle with power window to Power Window
  - Control Unit
  - Power Window to Control Unit
  - Sensor Devices
  - Power Window to Sensor Devices
  - Power Supply
  - Control Unit to Power Supply
  - Processor

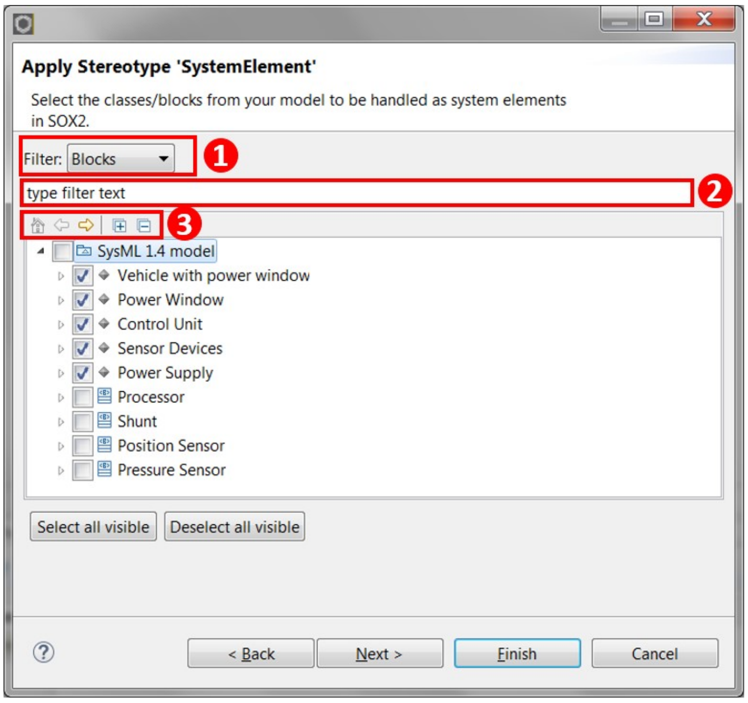
- The next page displays all issues (if any), such as modelling constructs not compliant to UML2/SysML standard or references to data not part of the import file. The issues are sorted by their type and the type of model element/property where it occurred.


**Diagnosis**

The following issues were found in the input data during import:

No issues found.

- The next three pages support to apply the SOX-specific stereotypes to the imported models elements. There is a page each to apply the stereotype *SystemElement*, *Function*, and *Malfunction*. A stereotype is applied by just checking the box next to the model element it should be applied to. Model elements where stereotype application is invalid are displayed in gray color and cannot be selected (for instance, the stereotype *SystemElement* can be applied to classes only). The filters on top of the page (1) can be used to filter for a model element type, e.g., all classes or blocks only. In addition, it is possible to filter the model elements name for a given text or text fragment (2). The model elements resulting from the chosen filters are shown together with their containers (e.g., containing package). Finally, the toolbar buttons (3) support to drill down into a sub-package (by selecting the package and using the arrow buttons) and to collapse/expand all elements that apply to the chosen filters.


Apply Stereotype 'SystemElement'

Select the classes/blocks from your model to be handled as system elements in SOX2.

Filter: Blocks

type filter text

- [ ] SysML 1.4 model
  - [✓] Vehicle with power window
  - [✓] Power Window
  - [✓] Control Unit
  - [✓] Sensor Devices
  - [✓] Power Supply
  - [ ] Processor
  - [ ] Shunt
  - [ ] Position Sensor
  - [ ] Pressure Sensor

The *Select all visible* and *Deselect all visible* support to select all elements that are visible, i.e., adhere to the chosen filters and whose container is not collapsed. For instance, to apply the stereotype *SystemElement* to all Blocks contained in a specific package and all its subpackages, set the filter to *Blocks*, select the respective package and click the *drill down* (rightarrow) button, click the *expand all* button, and finally click the *Select all visible button*. • After applying stereotypes as desired, click the *Finish* button and the model elements (including the selected stereotype applications) will be added to the target package in your existing project.

# 21. Known bugs and Limitations in the Release

This section lists known limitations in the current release.

## Server

- Multiple users working on the same project in parallel is supported but specific cases can cause the application to stop working so that a restart is required.

## Workbench

- History information in History view and Suspect Marker is currently supported only for objects that have a corresponding representation in SysML.

## FTA

- FTA Variants currently not working.

## Deleting objects

- Rare appearance of possible error messages due to deleting of elements. Elements will be deleted correctly but a restart may be required in this case.

# 22. License Agreement

IMPORTANT – READ CAREFULLY BEFORE USING – SafetyOffice X2 (SOX) - INDICATES YOUR ACCEPTANCE OF THIS AGREEMENT.

## 1. SCOPE OF APPLICATION

a. This End-User License Agreement ("EULA") is a legal agreement between the client („you“), i. e. a natural or legal entity or a partnership with legal capacity that acts upon conclusion of a legal transaction in exercise of its commercial or independent professional activities, and EnCo Software GmbH („we“) for the acquired software and/or embedded appliance software that accompanies this EULA, which includes computer software and may include associated media, printed materials, "online" or electronic documentation, and Internet-based services ("Software").

b. This EULA also applies to all updates and upgrades insofar as we make them available to the client after the installation of software updates. Future releases of the Software may warrant amendments to this EULA, so an amendment or addendum to this EULA may accompany the Software. EnCo Software GmbH may add to, change or remove any part, term, or Any such additions, changes, or removals shall apply as soon as they are brought to your attention. By subsequently continuing condition seth forth in this EULA at any time without prior notice or liability to you. the use of the Software, you are indicating your acceptance thereto.

YOU AGREE TO ALL TERMS AND CONDITIONS OF THIS EULA BY INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE.
IF YOU DO NOT AGREE, ENCO SOFTWARE GMBH IS UNWILLING TO LICENSE THE SOFTWARE TO YOU AND MAY NOT INSTALL, COPY,
OR USE THE SOFTWARE; YOU MAY CONTACT ENCO SOFTWARE GMBH FOR IMFORMATION ON RETURNING YOUR SOFTWARE.

## 2. RESERVATION OF RIGHTS

Our software is protected by international copyright laws, contracts and other legislation.
Subject to the following provisions, we reserve all rights and claims to the software, including all copyrights, patents, trade and operating secrets, trademarks and other intellectual property rights. This EULA does not confer on the client an exclusive right of use or exploitation for the software. With the exception of the rights referred to in sections 3 and 4 below the client does not acquire any rights to the software.

## 3. SUBJECT OF THE LICENSE

a. Unless otherwise agreed in writing the contractual software is only the standard software that has not been developed or produced individually for the client's needs.

b. The software we have made available corresponds to the latest state of technology and conforms to the product information and specifications provided by us or by our sales partners, including the information in the user manuals. We cannot guarantee that the software under this agreement is suitable for purposes that go beyond the fulfillment of our contractual obligations.

c. The client agrees with us that according to the latest state of technology programming errors cannot be ruled out with 100% certainty despite exercising the utmost conscientiousness and diligence and that it is not possible to develop software that is completely error-free for all application conditions. THE SOFTWARE IS NOT FAULT-TOLERANT AND IS NOT
DESIGNED, MANUFACTURED OR INTENDED FOR USE OR RESALE AS ONLINE CONTROL EQUIPMENT IN HAZARDOUS ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE.

d. In the case of standard software of third-party manufacturers we supply the client with the manufacturer's original user documentation; we are not obliged to supply any additional documentation. On request the client shall have the right to inspect the original user documentation to be supplied prior to conclusion of the agreement.

## 4. RIGHTS OF USE

a. If part of the subject of the agreement is the supply of standard software of a third-party manufacturer, its terms and conditions of use apply: we only broker the license agreement that is concluded directly between the manufacturer and the client. These terms and conditions of use shall be made available to the client on request, and if requested also prior to conclusion of the license agreement.

b. The following terms and conditions of use shall only apply unless otherwise stipulated in the terms and conditions of use pursuant to a. above:

c. The client shall be given the indefinite, but non-transferable und non-exclusive right to use the software. Unless otherwise agreed in writing, the subject of the agreement is not the network license (multi-user license), but rather is restricted to the right of use for the individual hardware. When changing hardware the software on the previously used hardware must be deleted in full. Simultaneously storing, keeping or using the software on more than just one hardware unit is not permitted.

d. If part of the subject of the agreement is a network license, this right of use shall only apply to the agreed number of individual stations on the contractually specified local network. The client is obliged to take suitable measures to prevent any unauthorized use by third parties; branches, companies affiliated to the client as licensee, partners or spatially or organizationally separate facilities belonging to the same institution shall be deemed “third parties”.

e. Unless otherwise agreed in writing or in whole or in part for any purpose other than as expressly permitted under this EULA or expressly permitted by applicable law notwithstanding this limitation, the client does not have the power as licensee to copy, modify, publish, adapt, redistribute, reverse engineer, decompile, disassemble, attempt to derive or discover source code, or otherwise reduce to a human-perceivable form, or create derivative works of, the Software or text material surrendered to him (codes, documentation) himself or have it done by third parties. The rights of the client under section 69c No.3, 69d (2) and (3) and 69e of the German Copyright Law (UrhG) shall remain unaffected thereby.

f. If the client would like to reverse engineer, decompile or disassemble (hereinafter referred to as “decompilation”) the computer program in order to achieve interoperability with other computer programs pursuant to section 69e UrhG, the client must contact us prior to decompilation of the computer program and request provision of the information required to achieve such interoperability. If we provide this information regarding interoperability without negligently hesitating, the client is not entitled to decompile the computer program.

g. Unless the above license conditions specify otherwise, the resale, lease for purposes other than purchasing or the loan of the software as well as any transfer for independent use is permitted within the legal limits and only on the following cumulative, additional conditions:

- Name and address of the purchaser or user shall be communicated to us in writing,
- The purchaser has agreed to our license conditions as well as the license conditions of thirdparty manufacturers whose standard software is contained in the software and
- The client has deleted or destroyed all the remaining copies or elements of the software from his system and all external data carriers, including back-up copies, in such a manner that he has no remaining means of using the software or parts of it and this can be proven to us on request.

## 5. EXCLUDED SOFTWARE

a. Notwithstanding the foregoing limited license grant, you acknowledge that the Software includes software subject to other terms and conditions governing the use of such software other than this EULA (“Excluded Software”). Certain Excluded Software may be covered by open source software licenses (“Open Source Components”), which means any software licenses approved as open source licenses by the Open Source Initiative or any substantially similar licenses, including but not limited to any license that, as a condition of distribution of the software licensed under such license, may require that the distributor make the software available in source code format. Please contact EnCo Software for a list of applicable Excluded Software and the applicable terms and conditions governing its use. Such terms and conditions may be changed by the applicable third party at any time without liability to you. To the extent required by the licenses covering Open Source Components, the terms of such licenses will apply in lieu of the terms of this EULA. To the extent the terms of the licenses applicable to Open Source Components prohibit any of the restrictions in this EULA with respect to such Open Source Components, such restrictions will not apply to such Open Source Component. To the extent the terms of the licenses applicable to Open Source Components require EnCo Software GmbH to make an offer to provide source code in connection with the Software, such offer is hereby made.

b. In particular, parts of our software use the following Open Source components licensed under the GNU Lesser General Public License (LGPL), version 3.0:
- JFreeChart (http://www.jfree.org/jfreechart/)
- Glazed Lists (http://www.glazedlists.com/)
- JasperReports (https://community.jaspersoft.com/)

It is possible to replace those components by custom implementations in the folder “plugins” of the SOX installation. For details of LGPL see: https://www.gnu.org/licenses/lgpl-3.0-standalone.html

## 6. LIMITATION OF LIABILITY AND CONTRIBUTORY NEGLIGENCE

a. WE SHALL ONLY BE LIABLE FOR DAMAGE OTHER THAN RESULTING FROM LOSS OF LIFE, PHYSICAL INJURY OR ILLNESS IF THIS DAMAGE
IS BASED ON INTENTIONAL OR GROSSLY NEGLIGENT BEHAVIOR OR ON THE NEGLIGENT BREACH OF AN ESSENTIAL CONTRACTUAL OBLIGATION BY US OR OUR VICARIOUS AGENTS.

IN THE CASE OF SIMPLE NEGLIGENCE WE SHALL BE LIABLE IN THE EVENT OF THE BREACH OF AN ESSENTIAL CONTRACTUAL OBLIGATION ONLY
FOR SUCH DAMAGE THAT IS FORESEEABLE OR TYPICAL. A CONTRACTUALLY ESSENTIAL OBLIGATION IS AN OBLIGATION, THE FULFILLMENT OF
WHICH ALLOWS THE CONTRACT TO EVEN BE DULY IMPLEMENTED AT ALL AND ON THE OBSERVATION OF WHICH THE CLIENT CAN REGULARLY RELY.

CLAIMS ARISING FROM A GUARANTEE THAT WE HAVE GIVEN FOR THE QUALITY OF THE OBJECT OF PURCHASE AND THE PRODUCT LIABILITY
LAW SHALL REMAIN UNAFFECTED BY THE AFOREMENTIONED LIABILITY RESTRICTIONS.

b. WE SHALL BE LIABLE FOR THE LOSS OF DATA IN ACCORDANCE WITH THE PARAGRAPH ABOVE ONLY IF SUCH A LOSS WOULD NOT HAVE BEEN
AVOIDABLE THROUGH APPROPRIATE DATA BACK-UP MEASURES BY THE CLIENT. INADEQUATE DATA BACK-UP OCCURS IN PARTICULAR IF THE
CLIENT NEGLIGENTLY OMITS TO TAKE PRECAUTIONS THROUGH APPROPRIATE SECURITY MEASURES CORRESPONDING TO THE STATE OF
TECHNOLOGY AGAINST EXTERNAL EFFECTS, IN PARTICULAR AGAINST COMPUTER VIRUSES AND SUCH PHENOMENA THAT COULD PLACE INDIVIDUAL DATA OR AN ENTIRE DATA SET AT RISK.

## 7. COOPERATION OF THE CLIENT

a. The client shall be obliged to take suitable precautions to prevent unauthorized access to the software and the documentation by third parties. To do this, the client shall in particular store any original data carriers and/or back-up copies at a location protected against unauthorized access by third parties. The client shall advise its employees about the terms and conditions of this agreement that have to be complied with in a suitable manner.

b. The client is obliged to notify any detected faults in as detailed and reproducible way as possible. If the client makes a claim against us for subsequent performance and it transpires that a claim to subsequent performance does not exist (e.g. user error, improper handling, lack of a defect), the client shall reimburse us for all the costs incurred in connection with inspecting the goods and the subsequent performance unless he is not responsible for the recourse to us.

c. The client shall ensure that the requirements for the hardware and the other system environment communicated to him have been met prior to installation. He shall ensure that all his data have been backed up prior to each installation.

d. In order to mitigate any damages and costs arising out of or relating in any way to the subject matter of this agreement, the client is obliged to set up an appropriate data backup in regular and adequate intervals.

## 8. LIABILITY FOR DEFECTS AND OTHER DEFAULT IN PERFORMANCE

a. The client is obliged to inspect the surrendered software including the documentation within two weeks after delivery or download. This especially applies in regard to the completeness of the software and the documentation as well as the functionality of the underlying program functions. We must be notified of any defects that are detected in this connection or which are readily identifiable within four weeks after delivery or download in writing (section 126b of the German Civil Code (BGB)). A detailed description of the defects must be attached. Defects in the software or documentation that cannot be identified within the framework of the routine inspection pursuant to sentence 1 above must be confirmed within 2 weeks after their discovery in writing (section 126b BGB). If the duty to examine the goods and to give notice of defects is breached, the software is deemed approved in terms of the respective defect.

b. Technical data, specifications and performance figures in public statements, especially in advertising, are not a specification of the quality of the software. The functionality of software is based on the performance specification stipulated in the offer or the contract.

c. In the case of material defects in supplied standard third-party software and when calling in third parties for maintenance work, we are entitled, insofar as it has a debt-discharging effect for the purposes of remedying the defect or a replacement delivery, to assign our corresponding claims against our supplier, the manufacturer or other third parties to the client unless this would be unreasonable for the client. The aforementioned also applies if we have adapted, configured or otherwise modified the software or hardware to suit the client's needs unless the material defect has been caused by our performance.

d. In the event of the client's intervention in the software, especially the program code, which is not expressly permitted by the operating instructions or other instructions for use, the client shall not be entitled to any claims due to defects if the client does not demonstrate and prove that the defect is not based on the intervention.

e. The client's warranty claims due to defects in the object of purchase shall become time-barred in one year from the passage of risk. This provision does not include claims for damages, claims due to fraudulent concealment and claims arising from a guarantee that we have given for the quality of the goods. Likewise the right of recourse under section 478 BGB is excluded. The statutory limitation periods shall apply to these excluded claims.

## 9. MISCELLANEOUS

a. The place of performance is the registered office of our company in Munich, Germany.

b. These license conditions and all the legal relationships of the parties are subject to German substantive law.

The application of the UN Convention on Contracts for the International Sale of Goods (CISG) is excluded.

c. The exclusive place of jurisdiction for all disputes arising from this contractual relationship is the registered office of our company in Munich, Germany, but we are entitled to institute legal proceedings against the client at another legal place of jurisdiction.

d. If any part of this EULA is held invalid, illegal, or unenforceable, that provision shall be enforced to the maximum extent permissible so as to maintain the intent of this EULA, and the other parts will remain in full force and effect. The parties are then always respectively obliged to replace invalid provisions with valid provisions that come as close as possible to the aim or the invalid provisions.

--
EnCo Software GmbH
Lyonel-Feininger-Str. 26
80807 München

HR-Nr.: HRB 105310