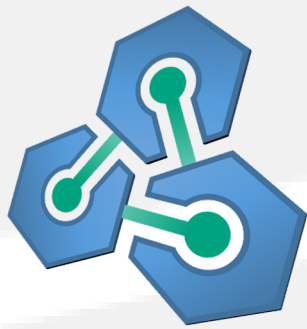




safeTbox

User Manual

V3.3

Fraunhofer IESE is an institute of the Fraunhofer-Gesellschaft.

The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competitive market position.

Fraunhofer IESE is directed by
Prof. Dr.-Ing. Peter Liggesmeyer

Fraunhofer-Platz 1
67663 Kaiserslautern
Germany

Contents

1	Introducing safeTbox	6
1.1	Target Audience	6
1.2	What is safeTbox?	6
1.3	Welcome Screen	7
1.4	Documentation and support	8
2	Installation	9
2.1	Installing the application	9
2.1.1	<i>System requirements</i>	9
2.1.2	<i>Installation steps</i>	9
2.2	Checking your installation	9
2.3	Licensing information	10
3	Getting Started	11
3.1	User interface	11
3.1.1	<i>Smart Menu</i>	11
3.1.2	<i>Extensions Menu</i>	11
3.1.3	<i>Settings Management</i>	12
3.1.4	<i>Hotkey Handlers</i>	13
3.1.5	<i>Application Console</i>	13
3.2	Creating and editing projects	13
3.2.1	<i>Create a diagram</i>	14
3.2.2	<i>Modeling an element</i>	14
3.2.3	<i>Modeling a connection</i>	15
4	Modeling Capabilities	16
4.1	Architecture Design	16
4.1.1	<i>Component Modeling</i>	16
4.1.2	<i>Component Modeling Constructs</i>	16
4.1.3	<i>Component Realization Views</i>	17
4.1.4	<i>Interface Formalization with Flow Type Systems</i>	19
4.1.5	<i>UML/SysML Support</i>	21
4.1.6	<i>Fault Model Integration</i>	21
4.2	Hazard and Risk Assessment (HARA)	22

4.2.1	<i>HARA – Creation</i>	22
4.2.2	<i>HARA- Sheets and their main functionalities</i>	22
4.2.3	<i>HARA - Common actions</i>	24
4.2.4	<i>HARA – Settings</i>	26
4.3	Fault Tree Analysis	29
4.3.1	<i>Component Fault Trees</i>	29
4.3.2	<i>Component Fault Trees Constructs</i>	29
4.3.3	<i>Component Fault Tree Realization Views</i>	31
4.3.4	<i>Common Cause Failures (CCF)</i>	32
4.3.5	<i>Interface Formalization - Generic Failure Type System</i>	33
4.3.6	<i>Interface Formalization - Flow Failure Type System</i>	35
4.3.7	<i>Visualization – Flattening a Component Fault Tree</i>	37
4.4	Systematic Theoretic Process Analysis (STPA)	38
4.4.1	<i>Definition of the purpose of the analysis</i>	39
4.4.2	<i>Construction of the control structure diagram</i>	39
4.4.3	<i>Identification of Unsafe Control Actions (UCA)</i>	41
4.4.4	<i>Identification of Causal Scenarios</i>	42
4.5	Safety Argumentation	43
4.5.1	<i>Goal Structuring Notation</i>	43
4.5.2	<i>Goal Structuring Notation Constructs</i>	43
5	Analysis Capabilities	47
5.1	Reliability Workbench Backend Configuration	47
5.2	Analysis overview	48
5.3	Analysis types	49
5.3.1	<i>Fault Trees - Qualitative Analysis</i>	49
5.3.2	<i>Fault Trees - Quantitative Analysis</i>	51
5.3.3	<i>Fault Trees - Analysis Context</i>	55
6	Usability Features	56
6.1	Activation/Deactivation of features	56
6.2	Backup Management	56
6.3	Diagram and Project History	58
6.4	Selection Synchronization	58
6.5	Find and Navigation	58
6.6	Drag and Drop	60
6.7	Deletion Mechanisms	60
6.8	Port Visibility Consistency	61

6.9	Properties Dialog.....	62
6.10	Properties View.....	64
6.11	Dynamic Trace Building	65
6.12	Diagram Layouting	68
6.13	Automatic Project Upgrade and Downgrade.....	71
6.14	Spell check	72
6.15	Default Visualization	74
6.16	Excel Exporter	75
7	Core Features	77
7.1	User Experience Improvement Program	77
7.1.1	<i>User Experience Reporter</i>	<i>77</i>
7.1.2	<i>Error Reporter</i>	<i>78</i>
8	Annexes.....	80
8.1	Annex 1	80

1 Introducing safeTbox

1.1 Target Audience

This document is targeted to attend end users of the tool, primarily embedded system architects/designers and safety engineers. Descriptions are tailored to users who are confident with model-driven approaches. Furthermore, we assume that users have some experience with Enterprise Architect (EA), but are not necessarily experts. From the technical point of view, we recommend that inexperienced EA modelers check the website of the vendor to get a better overview of the tool's modeling capabilities and UML based approaches.

1.2 What is safeTbox?

safeTbox stands for Safety Toolbox. It aims at supporting the development of software intensive embedded systems with safety critical characteristics. Most of the offered techniques can be used in different domains. However, the tool offers special support for the automotive domain and its associated standard ISO 26262. safeTbox integrates modeling as well as analysis capabilities. Thanks to its integrated approach, it also allows to achieve traceability between artifacts.

safeTbox supports primarily the following activities:

1. Architecture Design

safeTbox defines its own modeling language for the modular specification of embedded systems. It allows the definition of compositional and hierarchical models. safeTbox also supports the interaction with generic UML constructs and standardized languages like SysML.

2. Hazard and Risk Assessment (HARA)

safeTbox offers a table-based solution for performing the HARA. Traditionally this analysis is performed in excel, therefore this table-based solution allows an easy transition into the model-based integrated approach. This HARA implementation is ISO 26262 specific.

3. Fault Tree Analysis

Traditional fault trees as well as Component Fault Trees (CFT) are supported in safeTbox. CFTs is a component-oriented approach, in which architecture artifacts like functions or logical components are associated with modularized fault trees. Besides the modeling capabilities, safeTbox also provides the possibility to analyze these models qualitatively and quantitatively.

1 Introducing safeTbox

4. Systems Theoretic Process Analysis (STPA)

safeTbox offers a series of editors dedicated to support each of the steps defined in the STPA framework. The solution is flexible enough to allow the analysis of the system from different points of view. E.g. safety, security, performance, etc.

5. Safety Argumentation

As part of a typical safety lifecycle it is necessary to argue why the system will operate safely. In order to organize and structure those arguments in a modular way, safeTbox includes an implementation of the well-known technique "Goal Structuring Notation" (GSN). Which can be used to create safety concepts as well as safety cases.

1.3 Welcome Screen

After safeTbox has been installed and Enterprise Architect has been started, you will see the welcoming tab of safeTbox as shown in Figure 1. On this tab, you have the possibility to get to know safeTbox a bit better by accessing several sources of information, like examples, tutorials and the user manual. The tab will be always loaded at the start of EA, unless the user decides to check the option "Do not show at start". In order to see the tab again at EA start, the user will have to set this through the settings management, i.e. (after loading a project) Extensions Menu-> safeTbox -> settings -> application -> "show getting started tab at start".

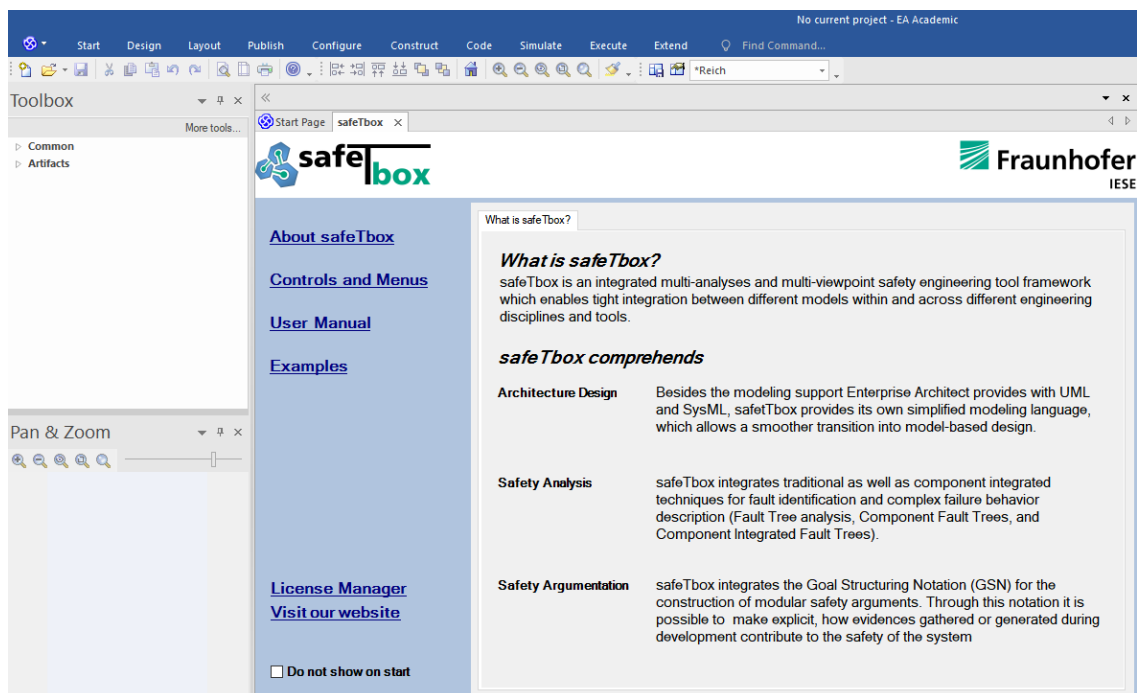


Figure 1 safeTbox Welcoming tab

1 Introducing safeTbox

1.4 Documentation and support

Besides this user manual, you might find further information about the tool and its capabilities in the:

1. [Website](#): general information of the tool, licensing, tutorials and training
2. Annex 1: research papers related to the methodologies implemented in the tool.

2 Installation

This section describes the steps required to install the safeTbox tool.

2.1 Installing the application

2.1.1 System requirements

Please make sure following requirements are satisfied before executing the installer of the tool.

1. Windows operating system (Windows 7 or later, Windows 10 recommended)
2. .NET framework 4.5 or higher
3. Enterprise Architect V14 or higher
4. Latest java runtime environment
5. An installation folder not requiring administrator rights
6. Any previous installation of the tool should have been removed (If your previous version was 1.0, invoking the installer automatically uninstalls the old version)
7. If you intend to perform safety analysis with FaultTree+ (Part of Isograph Reliability Workbench), make sure to have installed at least Reliability Workbench 13.0.2. as well as a dll API license.

2.1.2 Installation steps

1. Close Enterprise Architect if opened.
2. Uninstall any previous version of safeTbox.
3. Run the tool installer (e.g. EA_safeTBox_Setup_V3.0.msi) and follow the wizard.
 - a. Accept the license agreement.
 - b. Make sure that the requirements for a proper installation has been satisfied.
 - c. Decide if you want to participate in the improvement program
 - d. Select a directory in which no administrator rights are required.

2.2 Checking your installation

The About safeTbox dialog contains important information regarding the current installation of the tool. For example, the installation path and activated extensions. The dialog can be accessed under -> (Main Menu / EA context menu) -> Extension -> safeTbox -> About safeTbox, see Figure 2.

2 Installation

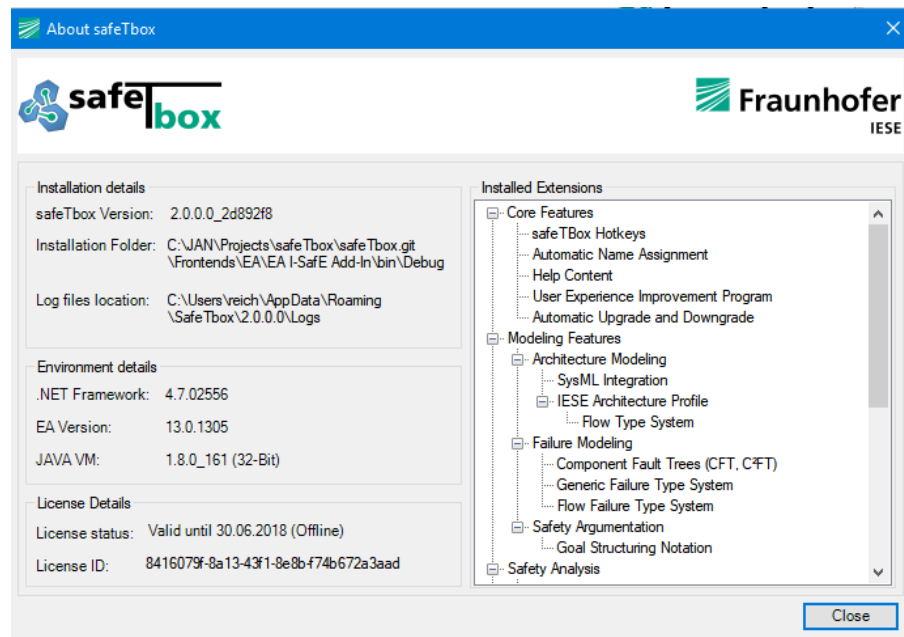


Figure 2 About safeTbox dialog

2.3 Licensing information

Currently safeTbox does not require any license as it is completely free.

3 Getting Started

This section describes shortly the primary ways in which the functionality of the tool can be accessed.

3.1 User interface

The functionality delivered with safeTbox can be accessed by one of these means:

3.1.1 Smart Menu

This is the primary way to trigger actions within safeTbox. The smart menu concentrates most of the functionality of safeTbox. It is built out of context-dependent as well as context-independent actions (e.g. "Settings Management"). The context-dependent actions are loaded according to the selected object. Once a selection is done (e.g. mouse click), the smart menu can be shown with following keyboard combination **Ctrl + Space** or **Ctrl + 1**. See Figure 3.

Ctrl + space

Ctrl + 1

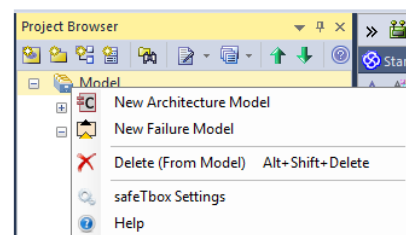


Figure 3 Smart menu example

3.1.2 Extensions Menu

It aggregates some of the context independent functionality of safeTbox. E.g. "Backup management". The aggregated actions can be triggered under -> (Main Menu / EA context menu) -> Extension -> safeTbox. See Figure 4.

3 Getting Started

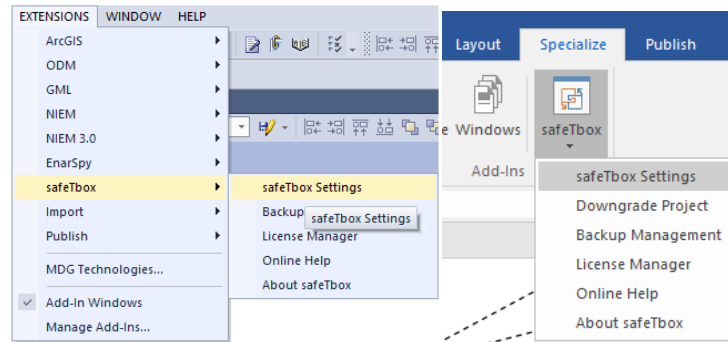


Figure 4 Extensions Menu example (Left – EA13, right EA14).

Other secondary means are related to features that are explained in the section Usability Features.

3.1.3 Settings Management

During its first run, safeTbox will create the settings file of the tool, which will be accessible through the “Smart Menu” -> safeTbox settings, or through the (Main menu/ EA context menu) -> Extensions -> safeTbox -> Settings.

safeTbox provides its own user settings in which the user might configure modeling as well as Usability Features. Depending on the deployed version of safeTbox the settings dialog might look different. See Figure 9.

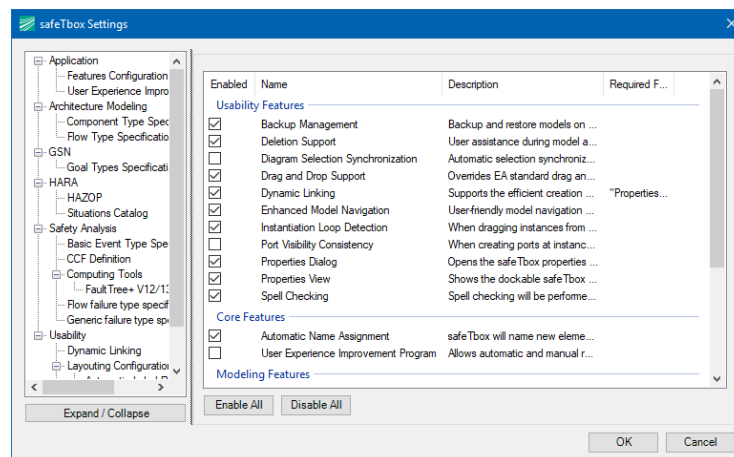


Figure 5 Settings dialog example

On the left side of the dialog the categories associated to the application or specific features are listed. On the right side of the dialog the individual settings can be set. If a feature has not been deployed with safeTbox, then the category associated with it will not be shown. For activating or deactivating safeTbox features see section Activation/Deactivation of features.

3 Getting Started

3.1.4 Hotkey Handlers

Hotkey handlers are keyboard shortcuts that allow to trigger actions in a faster way and without the need to use the “Smart Menu” or other EA context menus.

Following hotkey handlers are defined by default.

1. “Ctrl + 1” or “Ctrl + Space” => “Smart menu”
2. “Ctrl + Space + Shift”, “Ctrl + Space + Alt” => Execute last smart menu action
3. “Alt + Del” => Delete element from diagram
4. “Alt + Shift + Del” => Delete element from model

3.1.5 Application Console

The application console reflects the logging performed by safeTbox. This functionality has been developed for developers, because it contains primarily debug information. However, advanced users might find it useful. Moreover, in case something goes wrong with the tool, the logs presented in here can be sent to the developers so that an error can be more easily corrected in the next versions. This feature can be accessed through the short cut “Ctrl + Alt + D”, found. See Figure 6.

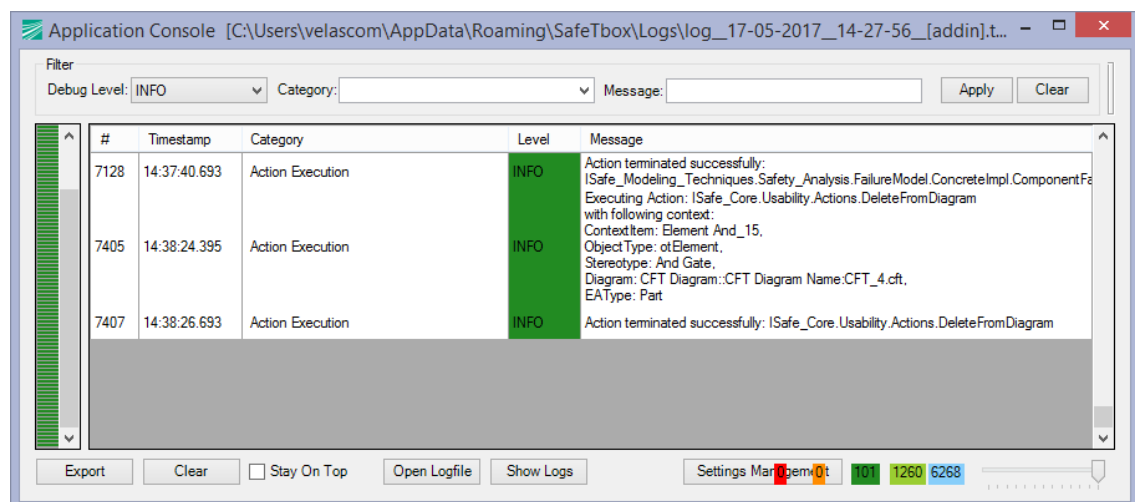


Figure 6 Application Console

3.2 Creating and editing projects

This section describes the different and generic ways of modeling objects within a diagram. A detailed description of each modeling technique is given in further sections.

3 Getting Started

3.2.1 Create a diagram

All modeling techniques in safeTbox are close related to some specific diagram type. E.g. if the user wants to model a component fault tree, a CFT-Diagram will be used. To facilitate the creation of the right diagram type, dedicated actions have been integrated in the "Smart menu". These actions will become available by calling the smart menu while having selected a "root node" (i.e. Model) or a "package" in the project browser. See Figure 3. Once a modeling technique has been selected and the diagram has been initialized, then it will be possible to add elements and connectors to the created diagrams by the usual means (e.g. toolbox).

3.2.2 Modeling an element

Modeling of elements within safeTbox is similar in all integrated modeling techniques. The user can achieve this by making use of any of these options:

1. Toolbox

Toolboxes will be loaded automatically according to the active diagram. In case the toolbox is not opened you could try to load it manually.

2. Smart Menu

The "Smart menu" facilitates the modeling process by offering creation actions to the most frequent elements within a diagram. This shall facilitate modeling, since connectors are added and layouted automatically. Simply select an element in a diagram and call the smart menu, "Ctrl + Space". In the creation section of the smart menu the elements that could be directly associated with the current selected element are offered. See Figure 7.

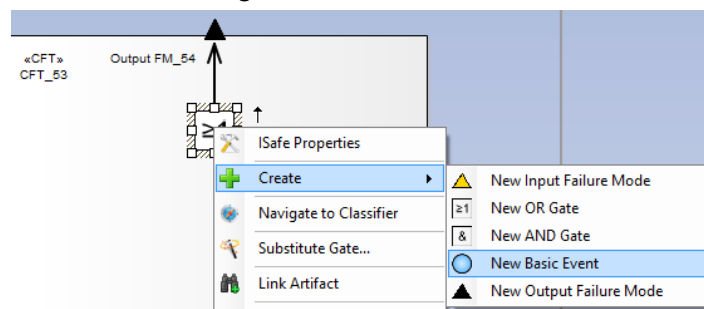


Figure 7 Modeling element through smart menu

3. EA Quick Menu

This context menu is provided by EA when selecting a modeling element in a diagram and typing "Space" or when holding the key "Ctrl + left mouse click". See Figure 8.

3 Getting Started

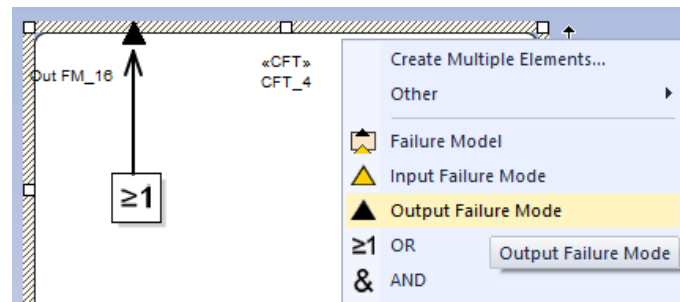


Figure 8 Modeling element through EA quick menu

4. Drag and drop

EA supports in general the modeling of existing elements by drag and drop. safeTbox facilitates this process and in some cases restricts or supports the modeling. This features can be deactivated through the settings.

3.2.3 Modeling a connection

Modeling of connections within safeTbox is similar in all integrated modeling techniques. The user can achieve this by making use of any of these options:

1. Toolbox

Similarly as for the modeling of elements.

2. Quicklinker

Quicklinker is a faster way of modeling connections between elements supported by EA. This feature is enabled by EA when selecting an element in a diagram, see Figure 9. safeTbox supported connectors appear always topmost and have the "(safeTbox)" tag in front of their name. We disregard the use of connections displayed in other sections, since these are EA default and they might give the model a semantic which is not part of the modeling technique and confuse other users.

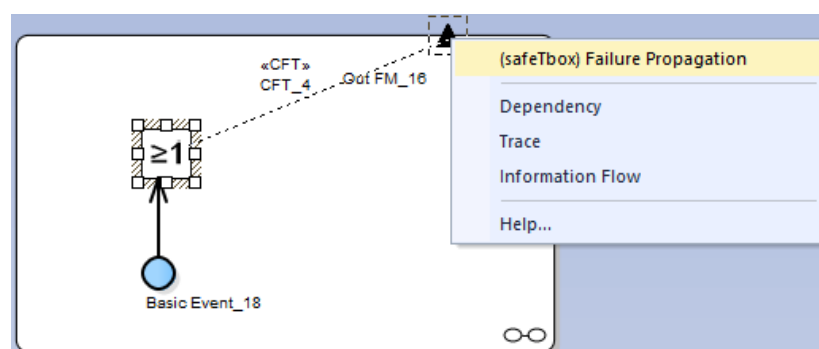


Figure 9 Quicklinker example

Once the connector has been added to the diagram, its attributes can be modified through the properties dialog. See section Properties Dialog.

4 Modeling Capabilities

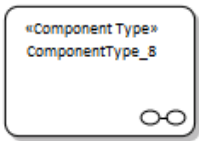
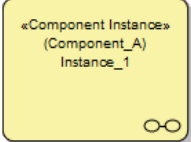
4.1 Architecture Design

The definition of the system's architecture is twofold supported in safeTbox. On the one hand, UML/SysML standard constructs are supported as provided by Enterprise Architect, which makes all modeling EA supported features available within safeTbox. On the other hand, safeTbox defines its own modeling approach that is better integrated with the safety analysis techniques than the SysML solution and which is better supported from the modeling perspective.

4.1.1 Component Modeling

The component modeling approach implemented in safeTbox works similarly as blocks within SysML. It serves to create modular, compositional and hierarchical system specifications. It is however simplified with respect to its syntax and semantics. While SysML provides a long range of port types, the component model approach considers only two types. This has been made simple, so that those getting into model-based design have more flexibility during modeling.

4.1.2 Component Modeling Constructs

Construct	Description	Toolbox icon	Diagram Shape
Component Type	A Component Type (often called classifier) can be used to depict functions, software or hardware components of an embedded system. It contains all other elements listed in this table. Moreover, it can be associated with a failure model (e.g. Component Fault Tree)		
Component Instance	It allows to instantiate a Component Type one or more times within a diagram or a project. It is therefore always linked to a Component Type, since it is a representation of it. Can be used to depict redundancies in the system.		

4 Modeling Capabilities

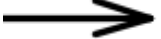
Inport	Represent the incoming interfaces of the component type. The concrete semantics should be specified by the modeler in its specific context. (The image on the right represents an inport annotated with a Flow type)		
Output	Represent the outgoing interfaces of the component type. Can be specified similarly as done for inports.		
Connector	These are directed connectors only meant to connect inports and outputs.		

Figure 10 shows a typical example of such component model.

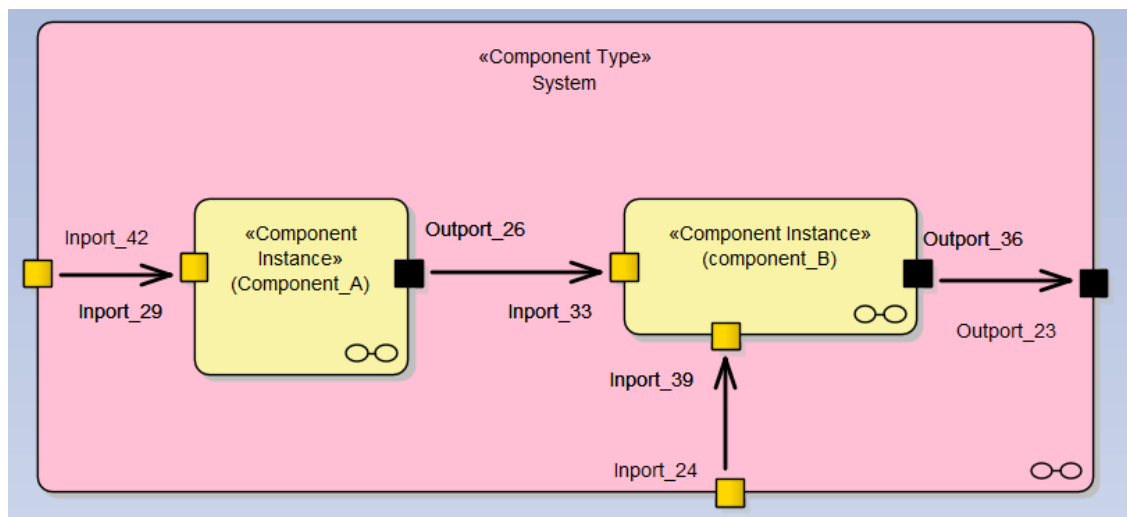


Figure 10 Component Modeling Example

4.1.3 Component Realization Views

When modeling components the engineer might want to capture different implementations of a component without creating an entirely new component. To enable this safeTbox supports multiple realization views, which allow depicting different aspects (e.g. variants, partial implantations) in different diagrams associated to the same component. Each component has at least one preferred realization view. Now, since a component might have several realization views, with the instantiation of a component the user should decide which of the existing realization views should be associated to the specific instance.

4 Modeling Capabilities

The realization view feature is enhanced by the Port Visibility Consistency feature, which will automatically synchronize ports of component instances based on the chosen realization view.

In Figure 11, actions related to the management of realization views are shown.

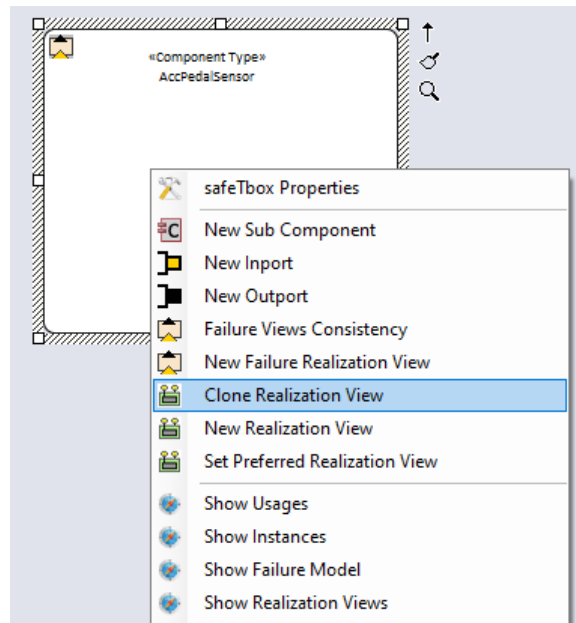


Figure 11 Smart menu for components

There are two options for cloning realization views, see Figure 12. “Only copy references” will simply reuse all elements from the realization view that is cloned so only existing elements are added to the new realization view. “Deep clone all elements on the realization view” will create a complete copy of every element on the cloned realization view and add them to the project browser. This is useful when the new realization view shall be completely separate from the old realization view.

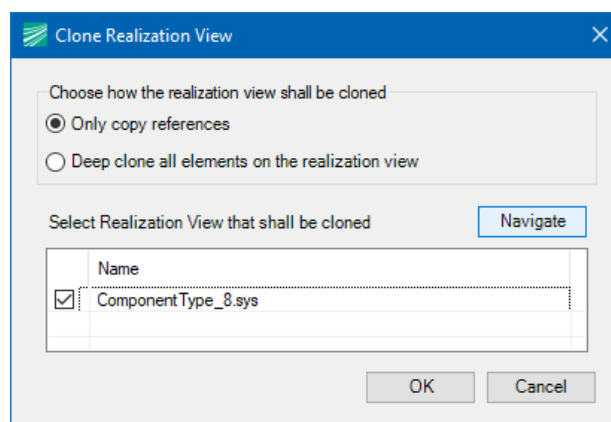


Figure 12 Cloning a realization view

4 Modeling Capabilities

4.1.4 Interface Formalization with Flow Type Systems

safeTbox allows the formalization of interfaces (inports and outports) in the component modeling approach via the specification of flow type systems. Flow Types provide the means for specifying and classifying the types of interaction components might have. Consider for example physical flows like fluids or logical flows like Data. In this solution the user has the possibility to create a hierarchy of flow types which can be used to depict abstract as well as concrete flow types.

When the feature is enabled, flow types can be defined through the dedicated Settings page, shown in the Figure 13: Architecture Modeling -> Flow Type Specification.

Each inport/outport is annotated with a default type "**NONE**" that the user can replace with the types defined in in the settings. The default type "**NONE**" cannot be edited, removed or have subtypes. Each flow type (except the default type) can have sub types that are represented as child nodes in the tree structure, See Figure 13.

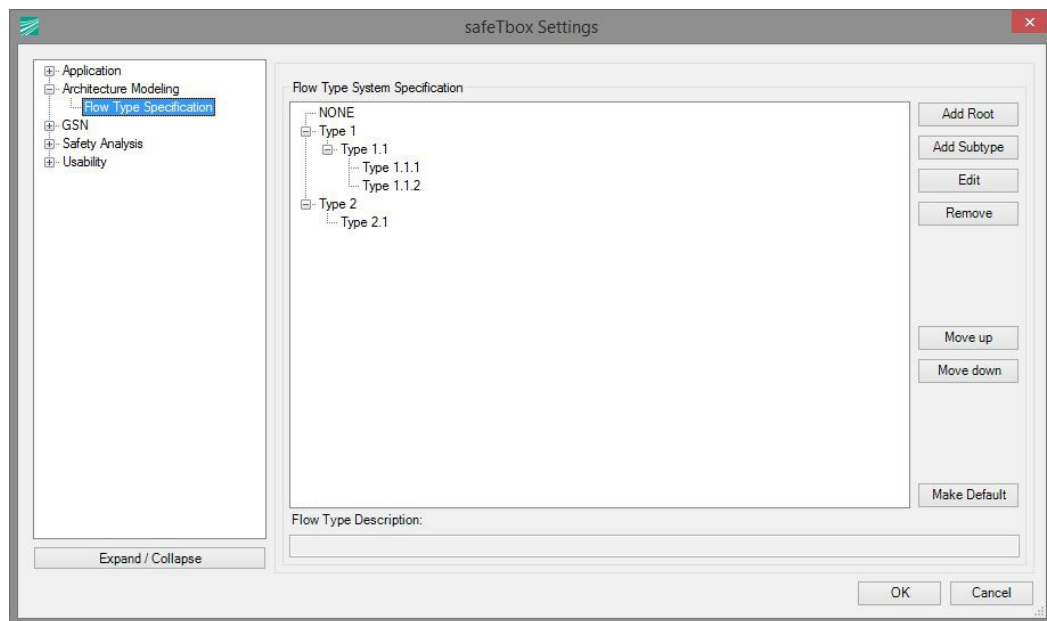


Figure 13 Flow Type specification page

When a Flow Type name is edited, all the modeling elements which have the Flow Type will be updated.

Note: Each type should have a unique name.

4 Modeling Capabilities

Flow types can be removed by using the **"Remove"** button. When a Flow Type is removed, all the modeling elements which have the Flow Type will be set to **"NONE"**.

Note: Removing a root node removes all the subtypes associated to that flow type.

By using the **"Make Default"** button the current flow type specification will be stored in such a way that for each new project the flow type specification will be automatically added.

Note: Once a configuration is set to default, it will be displayed in all the newly created projects. Existing projects will NOT have their configurations overridden. For that the import/export of settings might be of help. However, you need to consider that other settings might be overwritten.

If the flow type system feature is active, then when modeling an inport/outport, the **"Flow Type Selection"** dialog will be shown. This dialog can also be accessed through the properties view and properties dialog, see Figure 14).

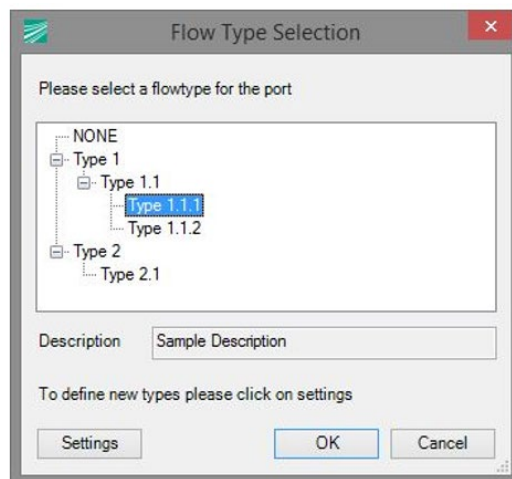


Figure 14 Flow Type - Selection Dialog

A Flow Type annotated port has a visual cue to separate it from the ports that don't have the annotation. See section 4.1.2.

The Flow Type of a port can be changed using the Properties View Window or the safeTbox Properties Window. The Flow Type attribute is shown in the safeTbox page under Interface Formalization group. (See Figure 15)

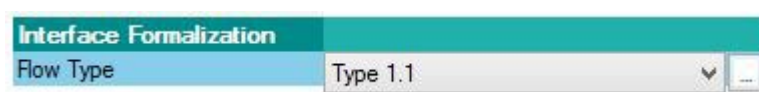


Figure 15 Properties View - Flow Type

4 Modeling Capabilities

The user can enable or disable the entire Flow Type System feature here, Settings management -> Features configuration -> Flow Type System.

Note: Disabling "Flow Type System" also disables the "Flow Failure Type System".

4.1.5 UML/SysML Support

Enterprise Architect supports UML and SysML itself. Its base version includes the UML constructs, while the SysML profile needs to be acquired separately. As an extension to that safeTbox supports the integration of SysML (Blocks) and UML (Activities) with the "Fault Tree Analysis" techniques. Similar as it occurs for the Component Modeling approach. See section "Fault Model Integration".

In this version, only few constructs from SysML are supported. Primarily Blocks, part properties and several port types (e.g. proxy ports, simple ports).

Figure 16 depicts a typical example of the kind of models we will expect to be created using SysML blocks, and for which the integration with safety analysis is supported.

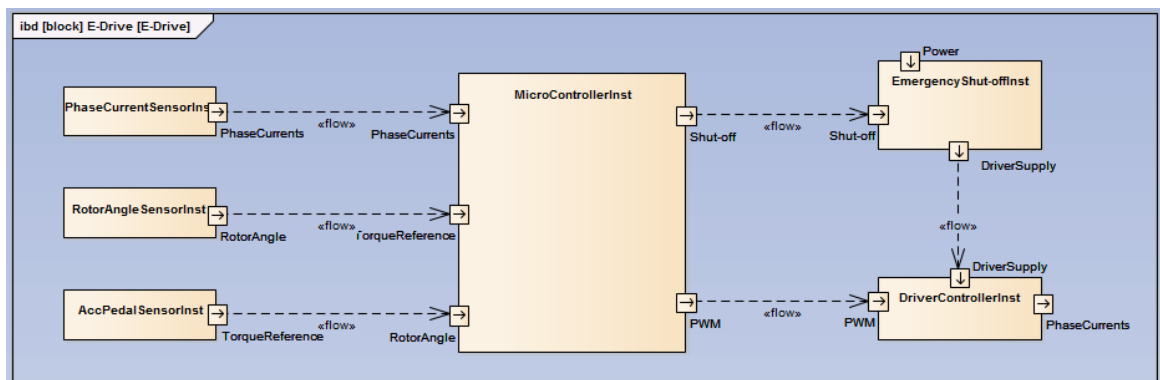


Figure 16 SysML blocks – Ibd example

4.1.6 Fault Model Integration

Architectural artifacts of the type/stereotype: "Component Type" and SysML "Blocks" can be integrated with fault models from the "Fault Tree Analysis" section. The idea of the integration with a fault model is to define the failure behavior of an entity, while making traceability explicitly between the interfaces of the given entity and the failure model, i.e. that ports are associated with failure modes. This concept fits very well to those systems descriptions based on data flows.

The user will find dedicated "Smart Menu" actions to add, navigate and remove the associations between the architectural artifacts and the failure models. Figure 17 exemplifies such integration.

4 Modeling Capabilities

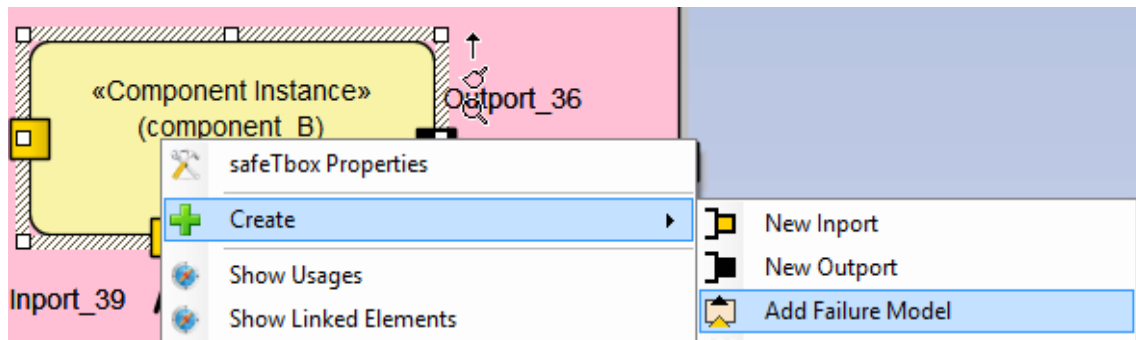


Figure 17 Component with integrated failure model

4.2 Hazard and Risk Assessment (HARA)

In this section, we describe how the user can create a table-based Hazard and Risk Analysis spreadsheet.

4.2.1 HARA – Creation

A HARA document can be created by using the smart menu. A new HARA package is created. If a HARA package exists already, it is added under that package.

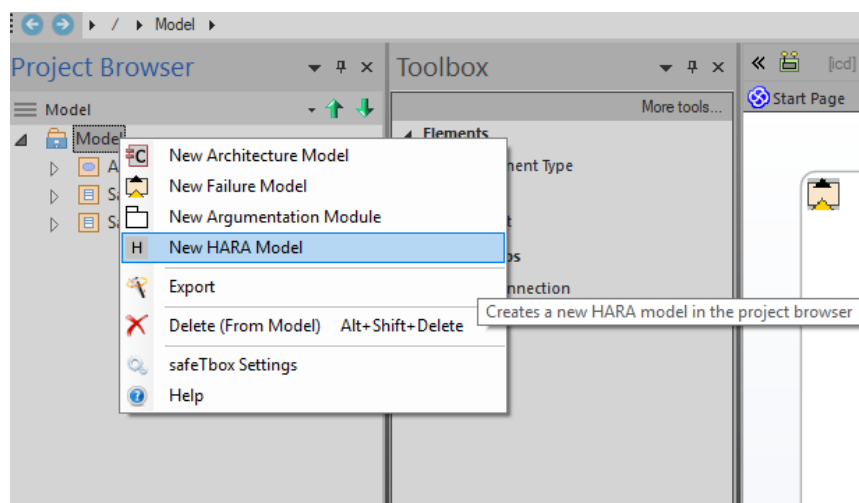


Figure 18 Creating HARA

A HARA spreadsheet contains 8 sheets. Each sheet provides specific actions for editing the document.

4.2.2 HARA- Sheets and their main functionalities

- **Cover sheet** – All the document related information can be filled here. The *Abbreviation* is used as a mask for the *Automatic ID assignment* feature mentioned in the next section

4 Modeling Capabilities

- **History sheet** – The user can add document history notes in this sheet.
- **Information sheet** – Contains relevant information which the user needs to work on this document. Is meant to be modified by the company for specific projects.
- **Function sheet** –
 - **Load existing component types**
Allows the user to load component types present in the project
- **FHA sheet** – It allows you to perform a Functional Hazard Analysis given the already loaded functions.
 - **Permutate functions with failure modes** - Permutates the functions present in the function sheet with the failure modes defined in HAZOP settings
 - **Create and Load existing Hazards** - Allows the user to define new hazards or load existing ones in the project. (Including hazards defined in another HARA document in the project)
 - **Tuple actions – Add/Remove/Reorder**- Allows the user to perform a set of tuple level actions on FHA entries or hazards.
- **Situation sheets** – It allows to define hazardous situations for driving and standing scenarios. The situations can be built based on the Driving and Standing situations catalogs, as configured in the settings dialog. The exposure parameter is defined in here.
 - **Update sheet** - Update the situation sheet with relevant Hazardous event entries.
- **Risk assessment sheet** – It allows to perform the risk assessment given the defined hazardous events. Severity and controllability parameters are together with their arguments specified here. This sheet also allows the user to define safety goals and document assumptions.
 - **Update sheet** - Update the sheet with relevant Hazardous event entries.
 - **Cell actions** -
 - **Add safety goal** – Create a new safety goal for the row. This unlocks the row by creating a new Safety goal with empty fields.
 - **Load existing safety goal** – Shows a list of existing safety goals that can be inserted into the row.
 - **Add assumption** – Similar to the Add safety goal function, allows the user to define a new assumption.
 - **Load assumption** – Shows a list of existing assumptions that can be inserted.

4 Modeling Capabilities

4.2.3 HARA - Common actions

- **Tuple / Column actions – Add/Remove/Reorder/Navigate** - Allows the user to perform a set of tuple level actions. A tuple level action is shown when the user clicks on the tuple number on the spreadsheet as shown in the figure below.

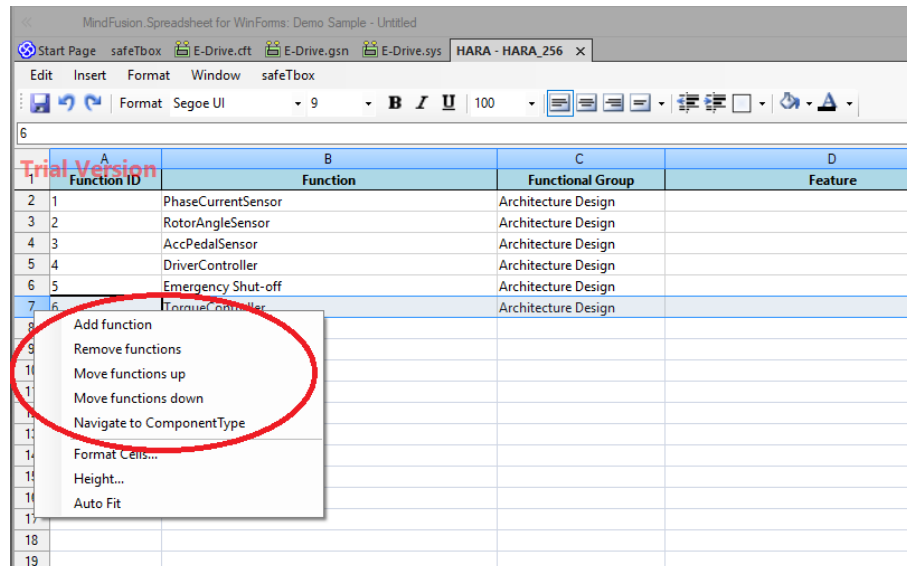


Figure 19 HARA sheet tuple actions

Column actions are shown when the user clicks on the column header as shown in the figure below.

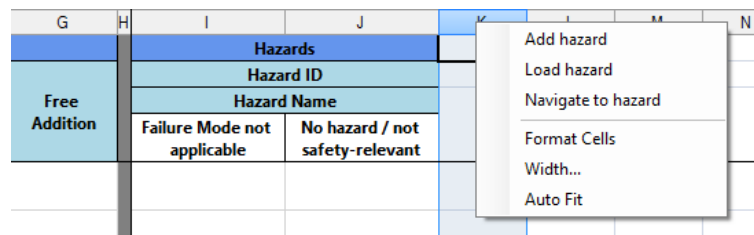


Figure 20 HARA column actions

- **Synchronization** - Naturally, Enterprise Architect does not support events history (undo/redo) properly. In order to enhance the usability of the HARA feature, we implemented this functionality. In order to achieve that, the HARA model is detached from the EA data model. Through this synchronization mechanism, it is then possible to make both data models consistent. The user can decide during the synchronization process if the values of the attributes are taken from the EA or the HARA data model.

4 Modeling Capabilities

- To Model – Synchronize changes made in HARA to EA Model elements.
- To HARA - Synchronize changes made in EA elements to HARA

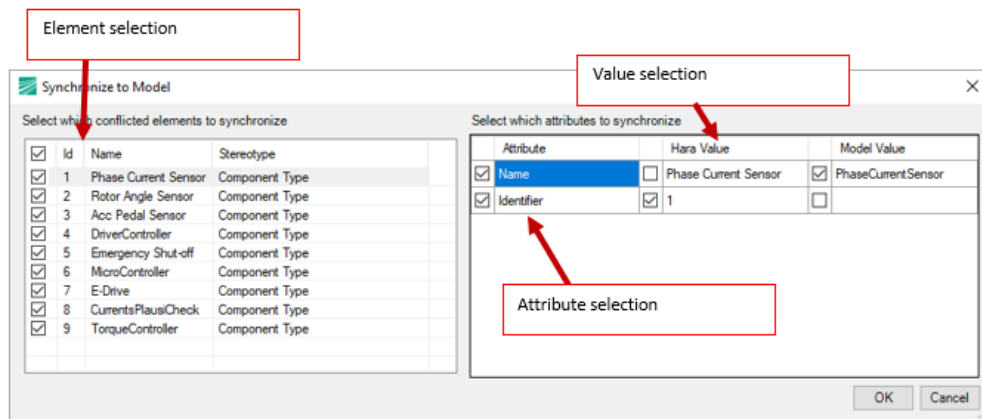


Figure 21 Synchronize to Model

The dialog shows a list of elements that have been changed and for each of them the list of attributes that have been modified, together with the value existing in the model and in the HARA table. The user can pick which value (HARA or Model) should be synchronized for each of the element present in the document.

- **Spell Check** – Performs a spell check on the document. Marks all cells that have spelling issues. Please check Section- 6.14 Spell check for further information with respect to the general spell functionality in the tool.
- **Automatic ID Assignment** – Uses the abbreviation in the cover sheet as a mask and auto fill the ID fields in FHA and the Situation sheets. (If enabled)
- **Validation** - Cells in HARA are validated based on a set of pre-defined rules. Each cell that fails its validation rule is set a background color based on the failed rule level. An appropriate message is annotated to this cell that describes the details of the rule failed.

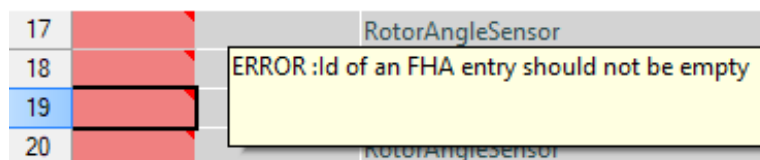


Figure 22 Validation – Error message

- **Cell Styles** - There are different types of cells in HARA and each cell is set a style based on some attributes. If a cell is read-only, it is set a gray background. If a cell has an error, it is set a red background as shown in the figure above. If a cell has a

4 Modeling Capabilities

warning message it is set a light salmon color. If a cell has failed spell check, the top border is set yellow. All of this are shown in the figure below.

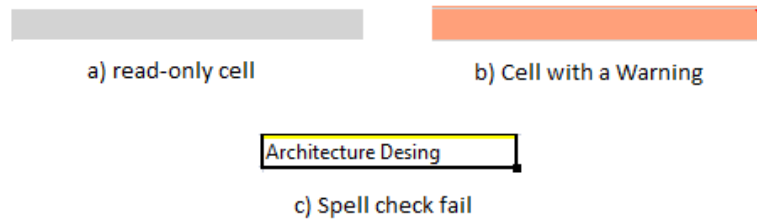


Figure 23 Different cell styles

- **History Mechanism** – SafeTbox allows the user to perform undo/redo operations on the HARA document. Certain operations such as changing the name of a function affects multiple sheets. These multiple changes can be undone/redone by a single undo/redo call.

4.2.4 HARA – Settings

Hazard and operability study (HAZOP) guidewords can be configured through the dedicated Settings page found under Settings -> HARA -> HAZOP

The defined HAZOP guidewords are available when using the Permutate functions with failure modes action in the FHA sheet.

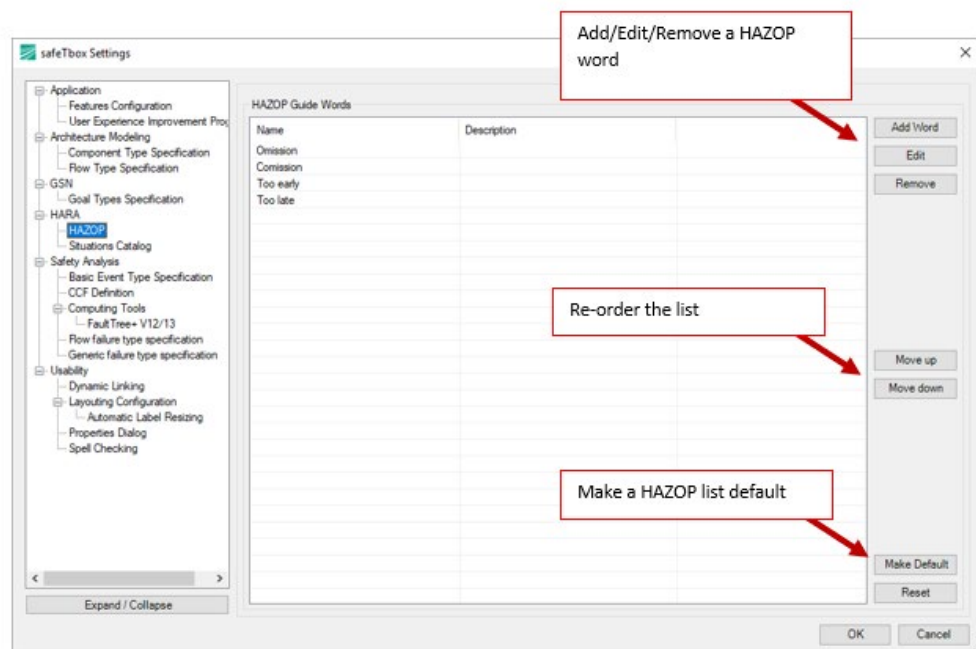


Figure 24 HAZOP settings

4 Modeling Capabilities

The situation catalog allows the user to define Categories and Situations for both the Standing and the Driving sheets.

Situation catalog can be configured through the dedicated Settings page found under Settings -> HARA -> Situation Catalog.

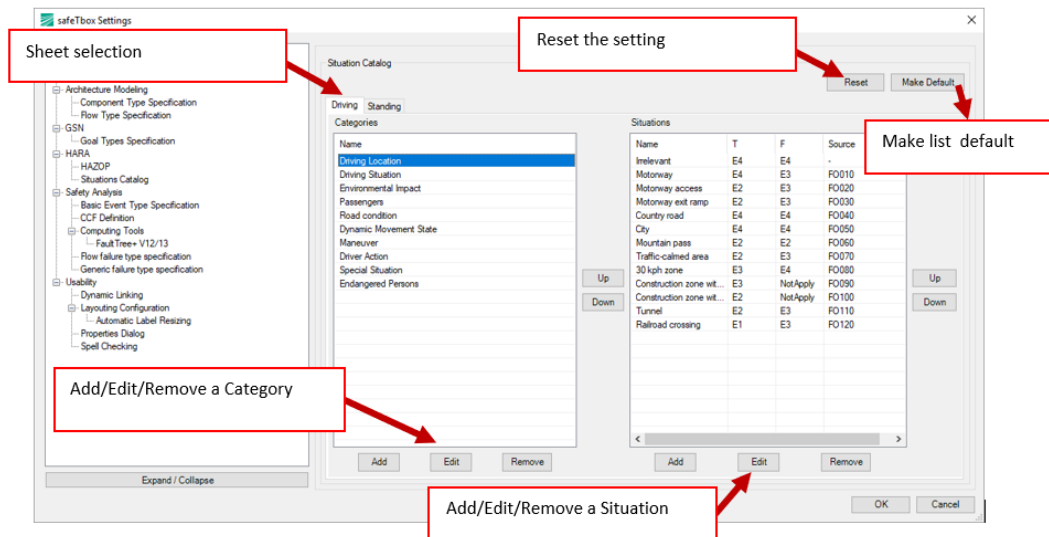


Figure 25 Situation Catalog

The Synchronization mechanism is triggered when any changes are made to the setting. The user can choose whether the changes made should be applied to HARA sheets or not. This is shown in the figure below. When the user clicks OK, the changes are then applied to the selected sheets and saved.

4 Modeling Capabilities

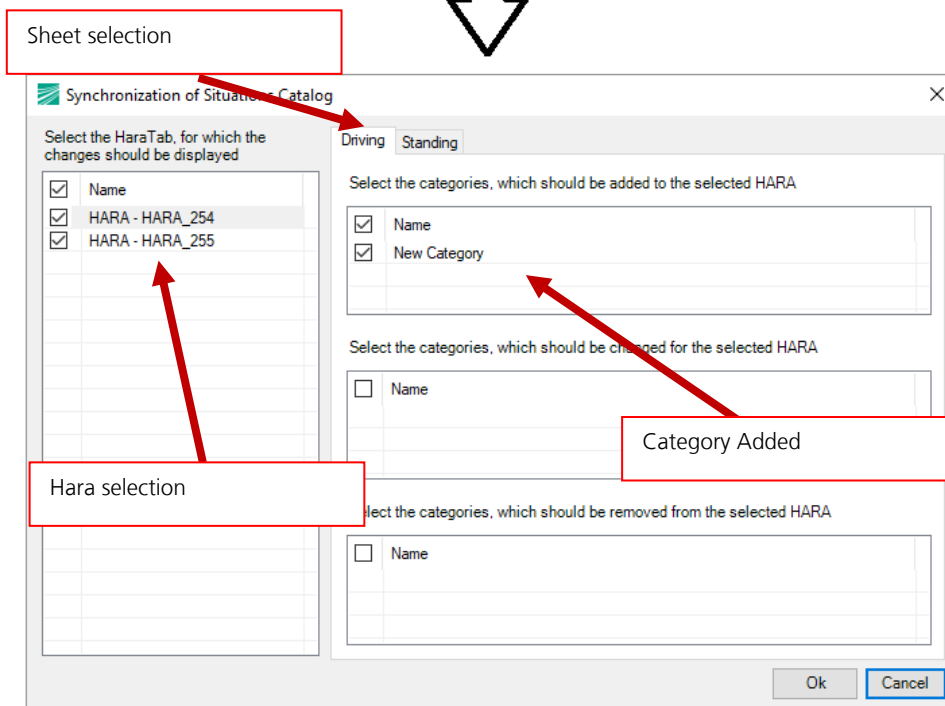
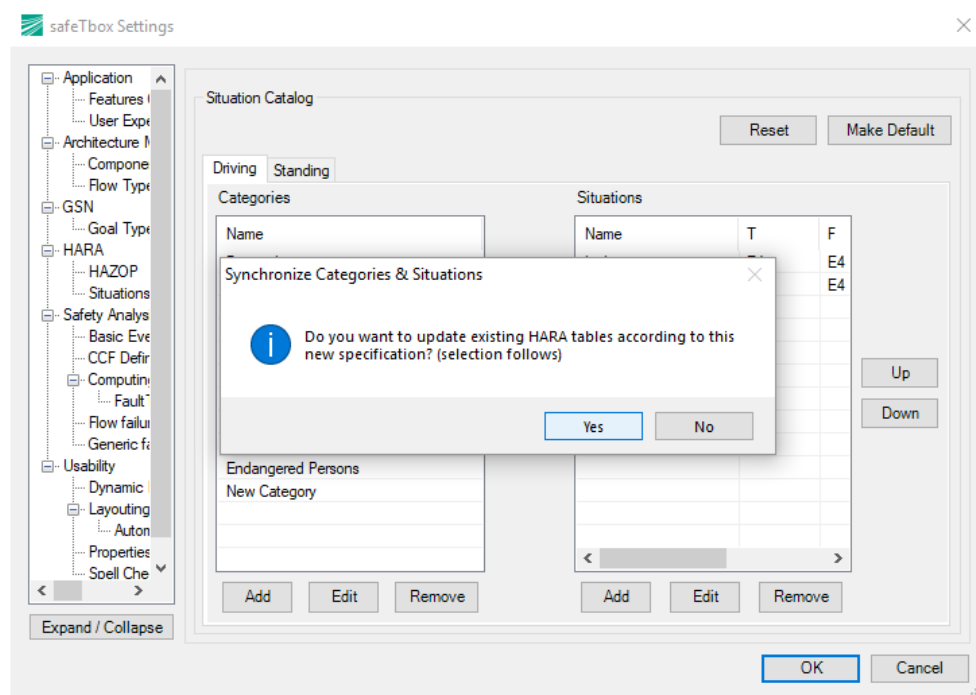


Figure 26 Synchronization for settings change

4 Modeling Capabilities

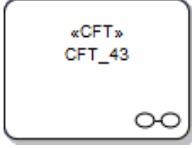
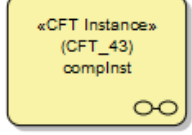
4.3 Fault Tree Analysis

In this section, we describe the means the user has to depict the failure behavior of a component using component fault trees. Note that fault trees is a primitive type that can be supported with a flat component fault tree.

4.3.1 Component Fault Trees

Component Fault Trees (CFTs) are the evolution of traditional Fault Tree Analysis, in which more modular and a better integration with the system design is achievable. CFTs provide the same constructs of a Fault Tree (i.e. top events (as Output Failure Modes), logical gates, and basic events). Moreover, it provides a further set of constructs that allows analyzing failures being propagated from other components (i.e. Input Failure Modes), as well as constructs to build hierarchical models (i.e. CFT Instance).

4.3.2 Component Fault Trees Constructs

Construct	Description	Toolbox icon	Diagram Shape
Component Fault Tree	This represents a modularized Fault Tree. It can contain all other constructs in this table. This technique can be used to depict the failure behavior of a Component Type or SysML block.		
Component Fault Tree instance	Similarly, as for Component Instances or part properties in the context of SysML. This entity allows to use a component fault tree in a specific context. i.e. in the specification of the failure behavior of other systems.		
Input Failure Mode	It represents the incoming interface of a component fault tree. This construct can be used to depict which external failure might have an impact in the definition of the failure behavior of one component. The white decoration on the right side indicates that the input failure mode has an associated formal failure type.		
Output Failure Mode	It represents the outgoing interface of a component fault tree. This construct can be used to depict the failures that a component exposes to its environment. The white decoration on the right side		

4 Modeling Capabilities

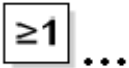
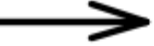
	indicates that the output failure mode has an associated formal failure type.		
Basic Event	It represents the basic causes that could lead to a failure of an entity. These are typically used to depict internal errors of components. It is possible though to define types. E.g. Security error		
Logical gate	OR, AND, XOR, NOT, and VOTE gates are considered within this category. These gates will allow to depict the failure logic within a component fault tree	$\geq 1 \ \& \neq$ $=1 \ V$	
Failure Propagation	It allows to define the failure logic from the internal point of view of a component fault tree by connecting logical gates, basic events, input and output failure modes. From the external point of view, it will help depicting the failure propagation between components, namely how outgoing failures from one component are interpreted as input failures by other components.		
Port Mapping	Whenever a component fault tree has been associated with component type or SysML block, there will be the possibility to define which interfaces from the design relate to which failure modes of the component fault tree. This type of connector has been conceived to define clearly the trace between these types of elements.		

Figure 27 shows a typical example of such Component Fault Tree.

4 Modeling Capabilities

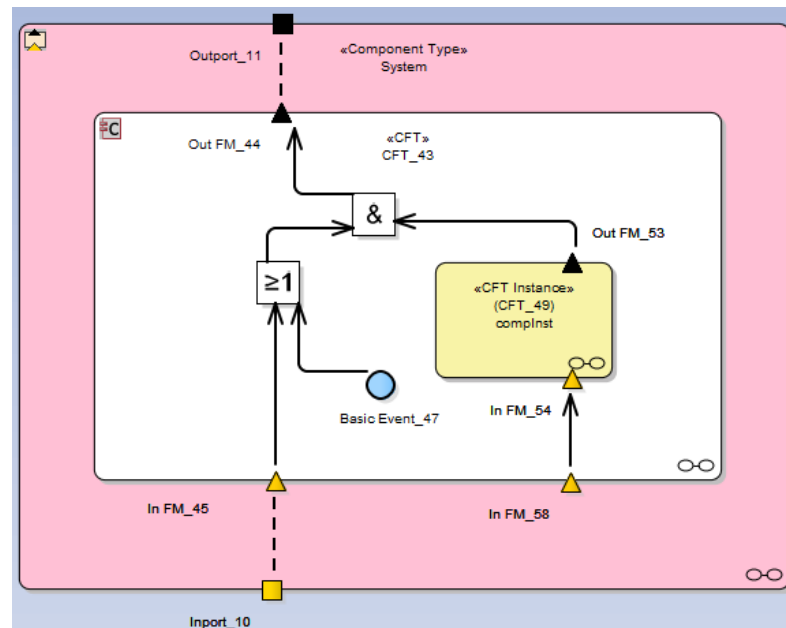


Figure 27 Component Fault Tree Example

4.3.3 Component Fault Tree Realization Views

Similar to regular safeTbox Component Types, CFTs can have multiple Realization Views. With them it is possible to define multiple failure views for the same CFT. This allows to create different versions of a CFT for example to handle variants, or for handling the system complexity. A CFT Instance always uses a specific realization view when the safety analysis is performed, and the safety engineer can easily change the realization view to test how different safety mechanisms affect the analysis results.

In safeTbox, CFTs can be associated with Component Types. The goal in doing so is to perform the safety analysis having as basis the logical design of the system. As seen before, component types can have multiple realization views, as well as CFTs. In order to clarify how this realization views relate to each other and to keep them consistent with respect to the shown interfaces, safeTbox offers a special dialog, which allows defining the mapping between them. This can be done for a selected component through the action called "Failure Views Consistency", see Figure 28.

Note: The feature Port Consistency should be enabled in the settings

In general, multiple CFTs can be mapped to one Component Type but one CFT cannot be mapped to multiple Component Types. This is because ports defined at the component realization view should be kept consistent with the ports shown at the component integrated CFT. This shall allow the user to perform a completeness check with respect to the analysis.

4 Modeling Capabilities

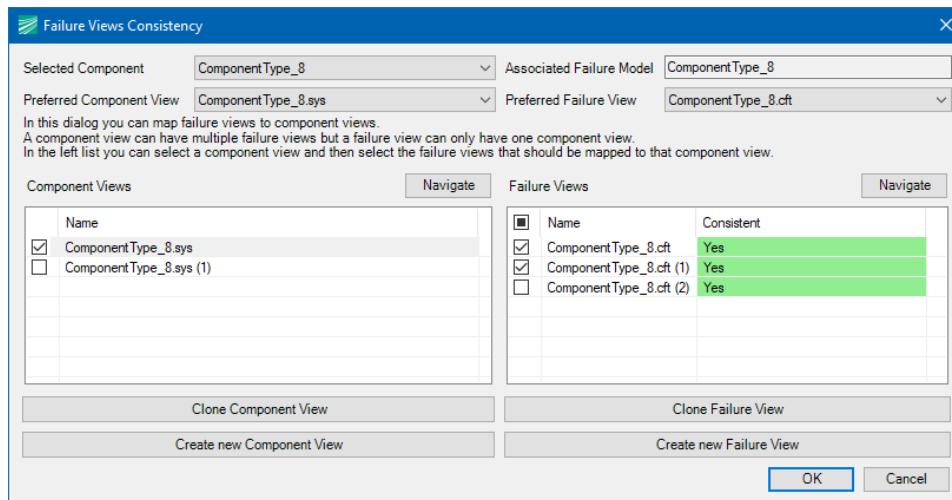


Figure 28 Failure View Consistency Dialog

4.3.4 Common Cause Failures (CCF)

SafeTbox provides modeling and analysis support for Common Cause Failures (CCF). CCF models can be created, edited and deleted in the CCF settings page as shown in Figure 29. Once a CCF model is defined, it can be applied to Basic Events within CFT diagrams. Within the settings page it is possible to see which basic events have been assigned to a CCF model and which CFT Instances are affected by it. I.e. Here the instances will be listed for which its associated realization view contains such basic event.

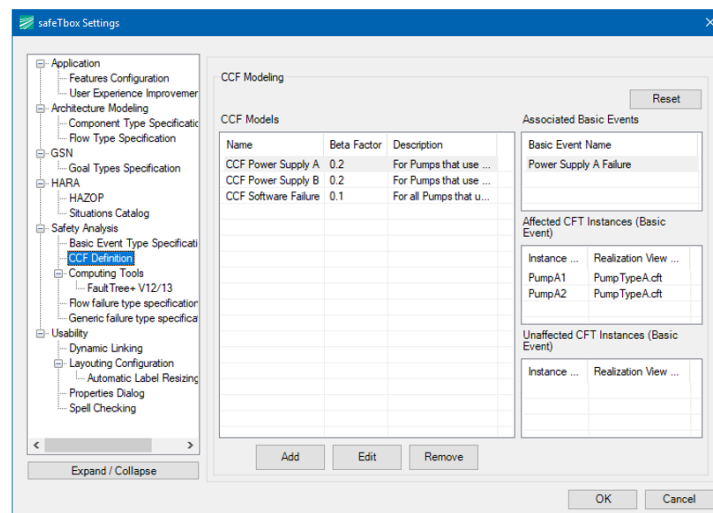


Figure 29 CCF Settings Page

The CCF model as well as the affected instances can be set through the action “Set CCF Model” see Figure 30 and Figure 31. Once a CCF model is associated to a basic

4 Modeling Capabilities

event, the basic event will display a small β (Beta) symbol on the diagram. CCFs will be considered during qualitative as well as in the quantitative analysis.

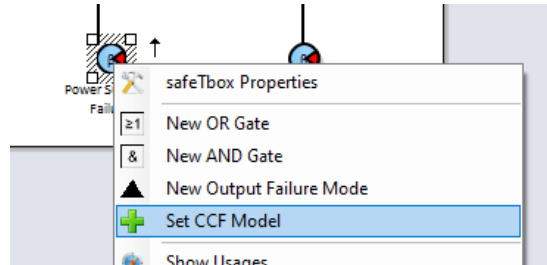


Figure 30 Set CCF Model Action

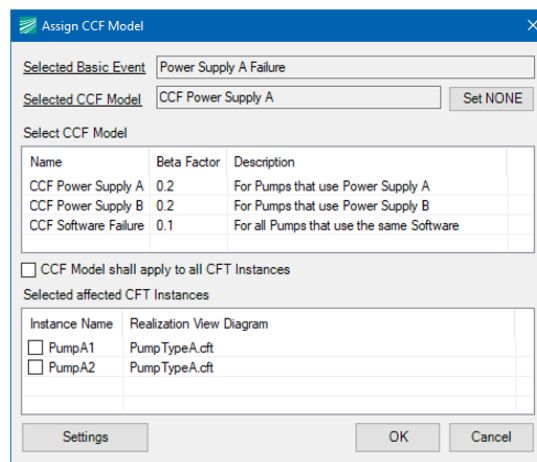


Figure 31 Assign CCF Model Dialog

4.3.5 Interface Formalization - Generic Failure Type System

safeTbox implements the "Generic Failure Type System" for the formalization of interfaces for CFTs. Specifically, for the input and output failure modes. Generic Failure Types can be used to annotate interface failure modes with guide words to standardize or specify the type of failure being propagated (e.g. HAZOP guidewords {Omission, commission...}). This can serve as well in a CFT to provide a machine-readable vocabulary for the interfaces to automate their composition (Introduced in Möhrle et al. 2005).

When the feature is enabled, Generic Failure Types can be defined through its dedicated settings page: Safety Analysis -> Generic Failure Type Specification.

A default type "NONE" exists for the user when he chooses not to annotate the element with any specific type. The default type "NONE" cannot be edited or removed. A set of pre-defined failure types (coming from HAZOP) are already configured. They can be modified if the user decides to. (See Figure 32)

4 Modeling Capabilities

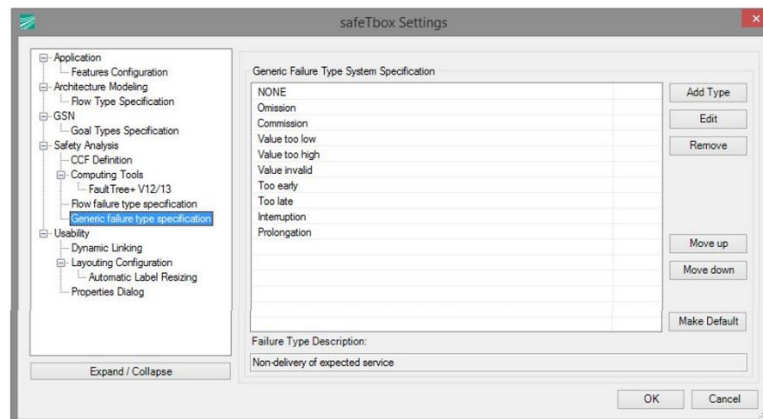


Figure 32 Generic Failure Type Specification

Note: This list is also used for the default flow failure type. (next sub-section)

Note: Each type should have a unique name.

Failure types can be removed by using the "Remove" button. When a Failure Type is removed, all the modeling elements which have the Failure Type will be set to "NONE".

Note: Once a configuration is set to default, it will be displayed in all the newly created projects. Existing projects will NOT have their configurations overridden.

When an input/output failure mode is modeled, a "Failure Type Selection" dialog is shown. (See Figure 33) This dialog shows the failure types defined in the previously mentioned "Generic Failure Type System". The user can define new types or edit existing types by clicking on the Settings button.

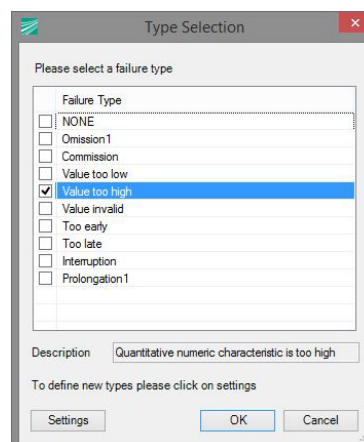


Figure 33 Generic Failure Type Selection Dialog

4 Modeling Capabilities

The Failure Type of a failure mode can be changed using the Properties View Window or the safeTbox Properties Window. The Failure Type attribute is shown in the safeTbox page under Interface Formalization group (See Figure 34). Moreover, a modeler will be able to define further details to the failure type, e.g. Value > 50 Kmh.

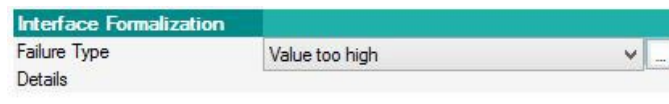


Figure 34 Properties View - Failure Type

The dropdown shows all the defined Failure Types as a list. The button next to the dropdown opens the Failure Type Selection dialog.

Note: When the feature is disabled, all the failure modes that have a Generic Failure Type are reset to "NONE".

4.3.6 Interface Formalization - Flow Failure Type System

In component integrated fault trees (C²FTs) input and output failure modes are supposed to be linked to inport and outports through the "port mapping" connector. This shall guarantee a proper traceability between the design and the failure models. In order to provide a more specific failure description, interface failure modes can be annotated with "Flow Failure Types". These are failure modes that can be defined specifically for Flow types. So, if you for example define the flow type: CAN_signal you can define the Flow Failure CAN_H_Interrupted.

Since Flow type systems can depict hierarchy of flows, the user will have to define flows relevant for the model and then the failure types associated to it, in order to get full use of the feature. The hierarchical relationship allows the sub flow types to inherit failure types of its parent flow type. If an inherited failure type is deemed not relevant to a sub flow type, it can be excluded (See Figure 35). Excluded failure types are not displayed in the selection.

The failure types can be defined for the flow types in Settings: Safety Analysis -> Flow Failure Type specification. (See Figure 35)

The default flow type contains the Generic failure types (See 4.2.3) as its failure types. Any changes to the Generic failure types are reflected to this type. This list can only be edited from the Generic Failure type specification settings page.

A new failure type can be associated to a flow type by selecting a flow type and clicking on "Add Type "button. A newly added failure type always has an Exclusion state - "Included". This means that the type is inherited to its sub types and the sub flow types have access to this failure type. Different flow types are allowed to have same failure types. This is however not true for two flow types with child parent relation.

4 Modeling Capabilities

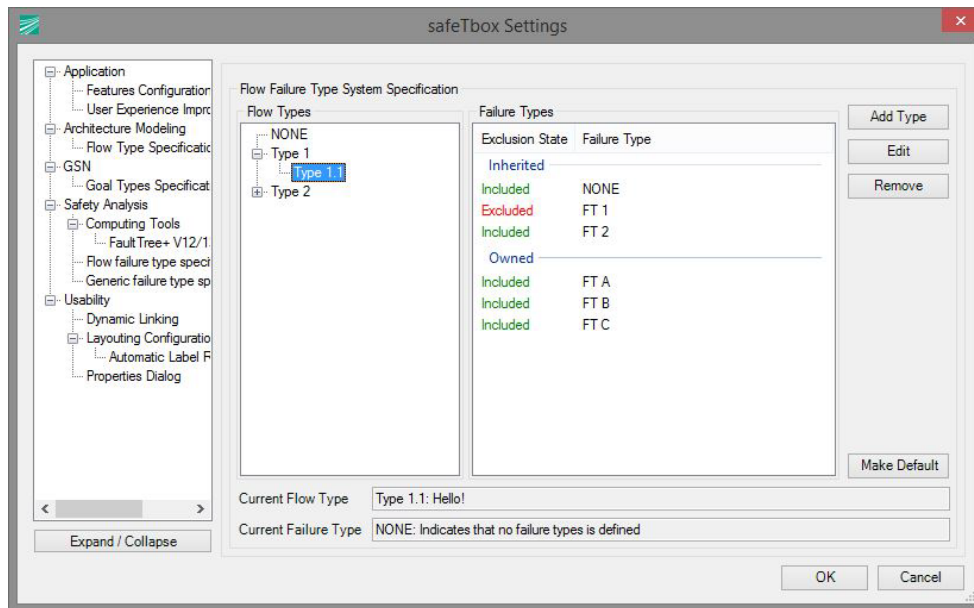


Figure 35 Flow Failure Type Specification

The Exclusion State can be changed by clicking on the State, which displays the drop down (only meant for inherited Failure Types) (See Figure 36). When a failure type is excluded at a parent Flow type, this failure type will also be excluded at all the sub types of the flow type. Automatically all the modelled interface failure modes associated to the excluded failure type will be set to the default type, i.e. NONE.

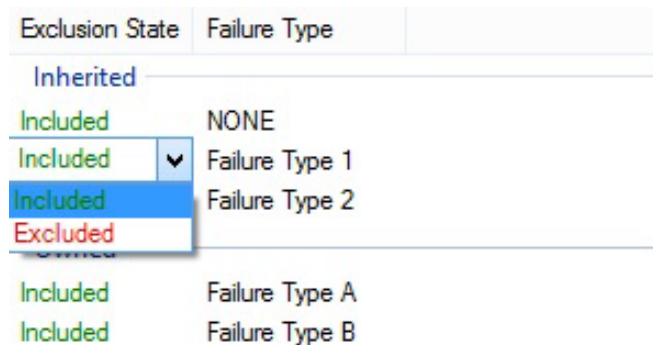


Figure 36 Exclusion State Change

When a failure type is removed, all the interface failure modes annotated with that failure type are set to the default type NONE. It is also removed in all the sub flow types, which inherited the type.

Note: When the feature is disabled, all the failure modes that have a Flow Failure Type set, are reset to "NONE".

4 Modeling Capabilities

4.3.7 Visualization – Flattening a Component Fault Tree

You can visualize a component fault tree as a flat fault tree by calling the action “Visualize flat CFT” in the smart menu of the CFT within a diagram.

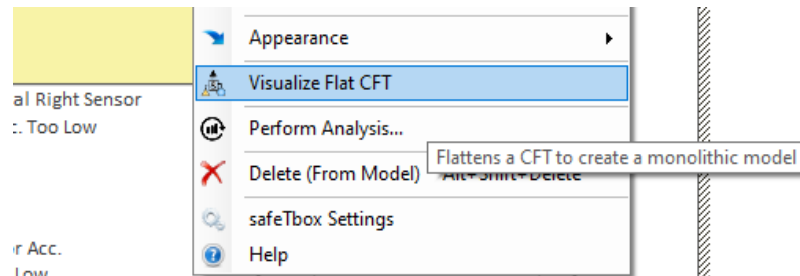


Figure 37 Visualize flat CFT action

This action will call a visualization dialog in which you will have the possibility to show the proxy failure modes. These are the input and output failure modes defined for each of the modularized fault trees. It will allow you to show / hide unconnected elements. These are elements that are not connected to the top events though failure propagation. A table on the side will list them as well. This will be handy for review purposes. Finally, you will have the possibility to show / hide the hierarchical names. By activating this feature, the names of the components will be added to the present modeling elements so that it is easier to recognize to which component they belong. This is shown in the following figure.

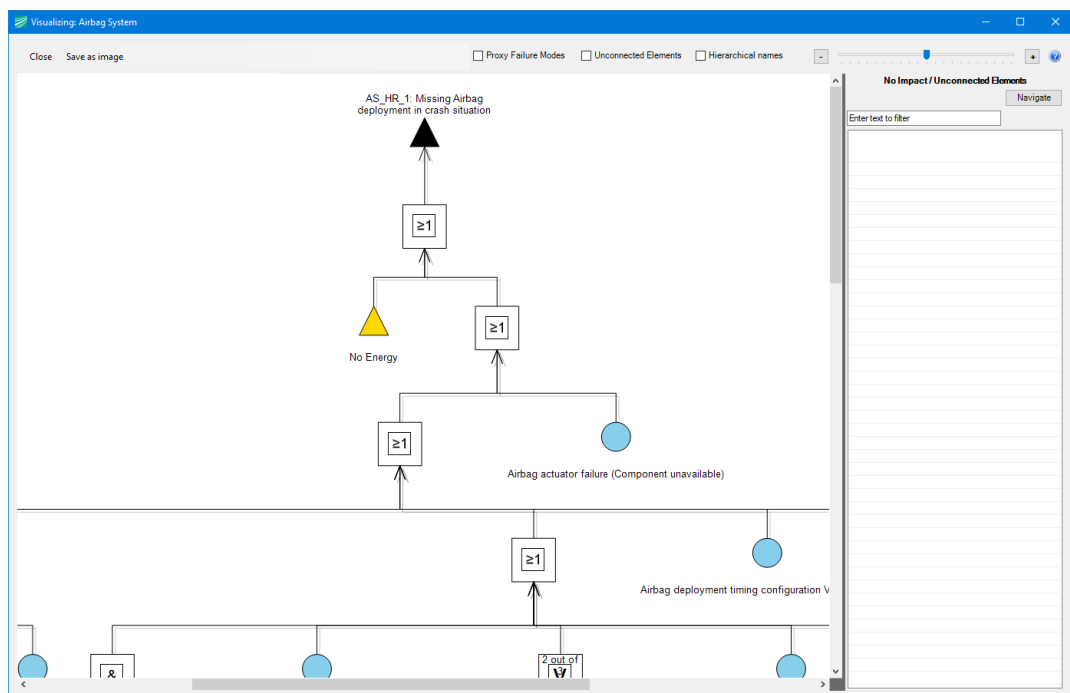


Figure 38 Fault CFT visualization dialog

4 Modeling Capabilities

4.4 Systematic Theoretic Process Analysis (STPA)

This section describes the tools available to the user to use STPA for safety analysis. Note that the editors can be tailored to analyze other system properties, e.g.

STPA consist mainly of 4 steps:

1. Definition of the purpose of the analysis
2. Construction of the control structure diagram
3. Identification of Unsafe Control Actions
4. Identification of Causal Scenarios

In safeTbox a dedicated editor has been created to aggregate these steps and to facilitate the management of the whole analysis process. The STPA Manager editor can be created for a folder as shown in the figure below.

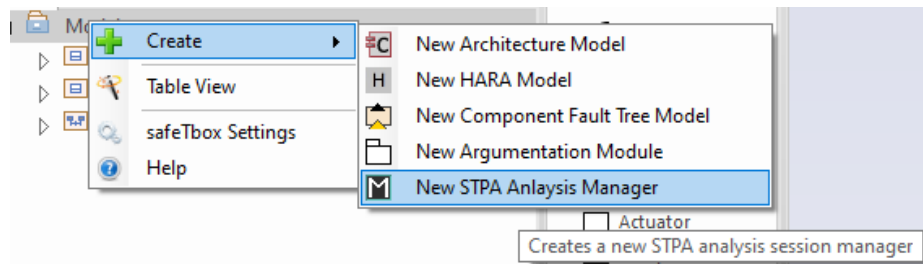


Figure 39 STPA manager creation action

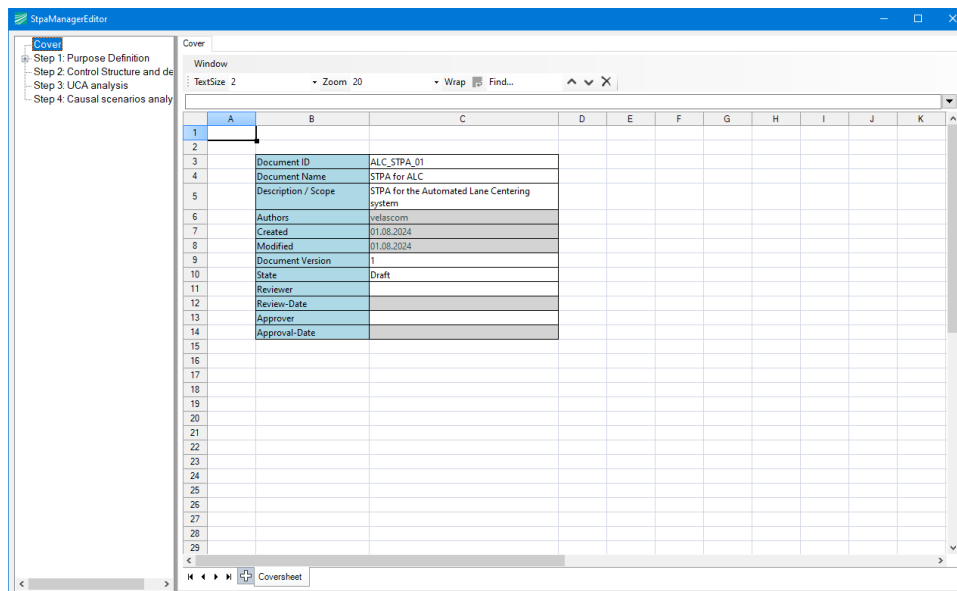


Figure 40 STPA manager dialog

4 Modeling Capabilities

4.4.1 Definition of the purpose of the analysis

Defining the purpose of the analysis is primarily done in the STPA Manager dialog. Note that the information in this section is not mandatory to proceed with the other steps in the process. However, it is recommended that this information is filled, as In STPA it is important to set appropriate boundaries and find a meaningful level of abstraction.

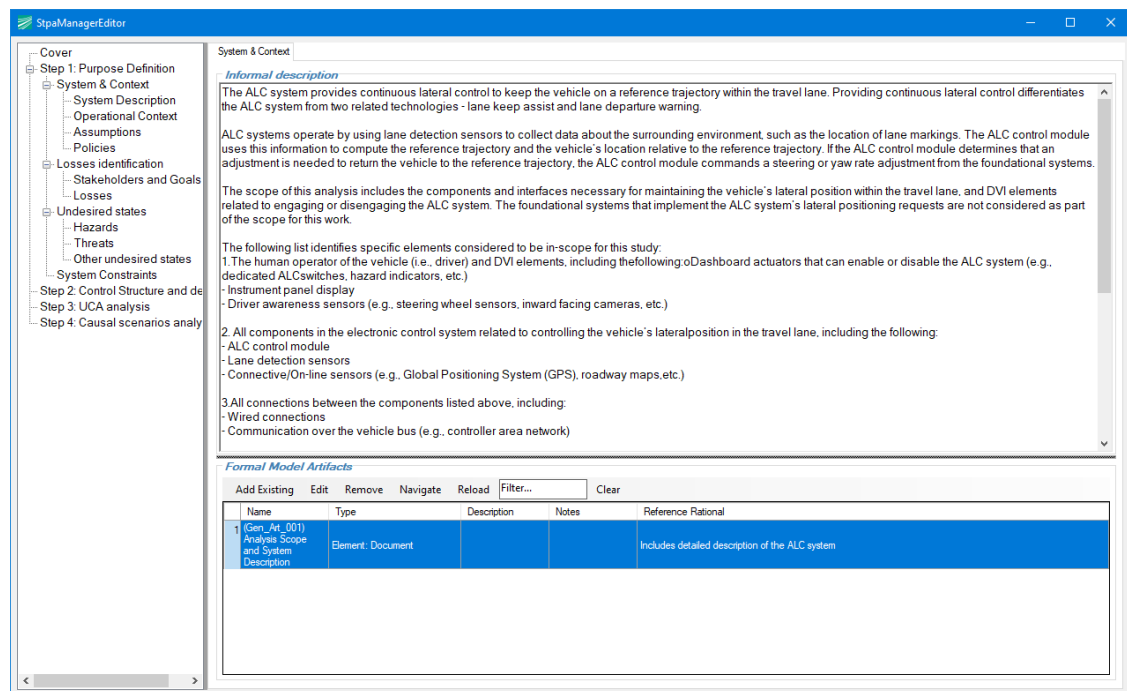


Figure 41 STPA – Defining the purpose of analysis

4.4.2 Construction of the control structure diagram

The control structure diagram can be created through the respective section in the STPA manager dialog. Since the analysis purpose might be unclear or too broad, the STPA manager will allow to create multiple control structure diagrams, see Figure 42. Moreover, by navigating to the selected control structure the associated diagram will be opened in EA as well as the control structure supporting editor, see Figure 43. This editor summarizes the modeled control actions and allow to define/ associated functional requirements.

4 Modeling Capabilities

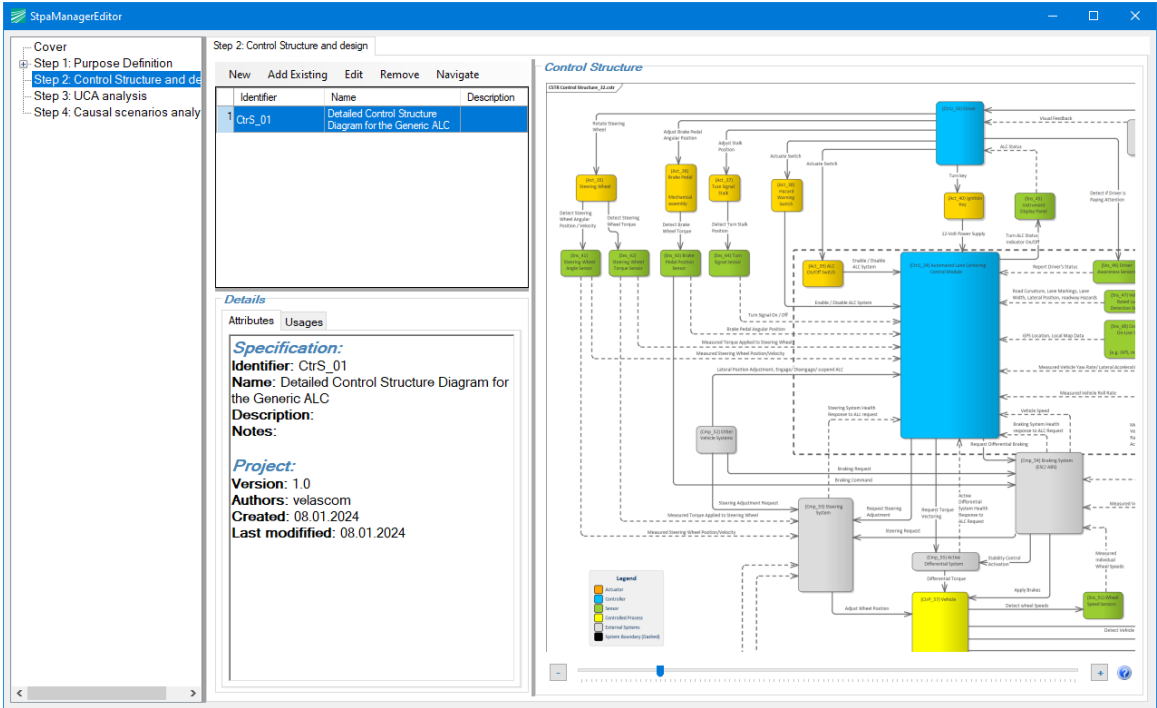


Figure 42 STPA – Creating a Control Structure Diagram

Control structure modeling support

Window

TextSize 2 Zoom 20 Wrap Find... ^ v X

Control Interaction ID

	A	B	C	D	E	F	G	H	I
1	Control Interaction ID	Control Interaction	Type	Source	Target	Control Interaction specific Requirements			
2	Fbk_102	Measured Vehicle Roll Rate	Feedback	(Sns_50) Roll Rate Sensor	(CtrU_34) Automated Lane Centering Control Module				
3	Fbk_104	ALC Status	Feedback	(Sns_45) Instrument Display Panel	(CtrU_33) Driver				
4	CtrA_105	Detect Vehicle Roll Rate	Control Action	(CtrP_57) Vehicle	(Sns_50) Roll Rate Sensor				
5	CtrA_106	Detect Vehicle Yaw Rate, Lateral Acceleration	Control Action	(CtrP_57) Vehicle	(Sns_49) Yaw Rate / Lateral Accel. Sensor				
6	CtrA_107	Sensory Feedback	Control Action	(CtrP_57) Vehicle	(CtrU_33) Driver				
7	CtrA_108	Detect wheel Speeds	Control Action	(CtrP_57) Vehicle	(Sns_51) Wheel Speed Sensors				
8	CtrA_109	Differential Torque	Control Action	(Cmp_55) Active Differential System	(CtrP_57) Vehicle				
9	Fbk_110	Measured Vehicle Yaw Rate/ Lateral Acceleration	Feedback	(Sns_49) Yaw Rate / Lateral Accel. Sensor	(Cmp_54) Braking System (ESC/ ABS)				
10	Fbk_113	Measured Vehicle Yaw Rate/ Lateral Acceleration	Feedback	(Sns_49) Yaw Rate / Lateral Accel. Sensor	(Cmp_53) Steering System				
11	Fbk_112	Measured Vehicle Roll Rate	Feedback	(Sns_50) Roll Rate Sensor	(Cmp_53) Steering System				
12	Fbk_111	Measured Vehicle Roll Rate	Feedback	(Sns_50) Roll Rate Sensor	(Cmp_54) Braking System (ESC/ ABS)				
13	Fbk_114	Vehicle Speed	Feedback	(Cmp_54) Braking System (ESC/ ABS)	(CtrU_34) Automated Lane Centering Control Module				
14	Fbk_115	Measured Individual Wheel Speeds	Feedback	(Sns_51) Wheel Speed Sensors	(Cmp_54) Braking System (ESC/ ABS)				
15	CtrA_116	Braking Request	Control Action	(Cmp_52) Other Vehicle Systems	(Cmp_54) Braking System (ESC/ ABS)				
16	CtrA_117	Braking Command	Control Action	(Sns_43) Brake Pedal Position Sensor	(Cmp_54) Braking System (ESC/ ABS)				
<	CtrA_120	Stability Control Activation	Control Action	(Cmp_54) Braking System	(Cmp_55) Active				

< > << >> + Coversheet Control structure in context Requirements Sheet

Figure 43 STPA – Control Structure supporting editor

4 Modeling Capabilities

4.4.3 Identification of Unsafe Control Actions (UCA)

One at least one control structure diagram exist it will be possible to start with the identification of unsafe control actions. A dedicated editor is used for this purpose. It can be initiated through the STPA manager in the respective section. Note that multiple UCA analyses can be started for the same control structure diagram. This might be desired if the complexity of the diagram is too high, for distributed analysis, or because of analysis of different concerns, e.g. safety / security.

Once the UCA analysis editor has been instantiated the user will have the possibility to add the control actions to be analyzed and proceed with the main parts of the analysis, i.e. with the identification of critical UCAs by associating these to hazards, threats or undesired states (generic form). As shown in following figure.

Unsafe Control Actions Analysis												
Unsafe/Threats control actions												
UCA Identifier	CS Identifier	Control Interaction	SourceName	TargetName	Guideword	UCA Description	Comments and arguments	Context variables and other effects	N/A or No Impact	Hazard Identifier	Hazard Name	
Uca_224	CS_149	ICL_149Request Steering Adjustment	Automated Lane Centering Control Module	Steering System	Executed incorrectly	Adjust Vehicle's Lateral Position in the δ Direction		- Direction of δ is away from the reference trajectory, δ in the opposite direction of δ, - Lateral Adjustment Request From Other Vehicle Systems, - Movement Relative to:	X	H01_01	Insufficient Lateral Adjustment Resulting in Lane-Roadway Departure While ALC is Engaged	
Uca_225	CS_149	ICL_149Request Steering Adjustment	Automated Lane Centering Control Module	Steering System	Too Soon	Adjust Vehicle's Lateral Position in the δ Direction		- Direction of δ is away from the reference trajectory, δ in the opposite direction of δ, - Lateral Adjustment Request From Other Vehicle Systems, - Movement Relative to:	X	H01_02	Excessive Lateral Adjustment Resulting in Lane-Roadway Departure While ALC is Engaged	
Uca_226	CS_149	ICL_149Request Steering Adjustment	Automated Lane Centering Control Module	Steering System	Too Late	Adjust Vehicle's Lateral Position in the δ Direction		- Direction of δ is away from the reference trajectory, δ in the opposite direction of δ, - Lateral Adjustment Request From Other Vehicle Systems, - Movement Relative to:	X	H01_03	Unexpected Loss of ALC	
Uca_227	CS_149	ICL_149Request Steering Adjustment	Automated Lane Centering Control Module	Steering System	Not Provided	Adjust Vehicle's Lateral Position in the δ Direction		- Direction of δ is away from the reference trajectory, δ in the opposite direction of δ, - Lateral Adjustment Request From Other Vehicle Systems, - Movement Relative to:		H01_04	Improper Transition of Control Between the Driver and ALC System	
Uca_228	CS_149	ICL_149Request Steering Adjustment	Automated Lane Centering Control Module	Steering System	Unintended Provided	Adjust Vehicle's Lateral Position in the δ Direction		- Direction of δ is away from the reference trajectory, δ in the opposite direction of δ, - Lateral Adjustment Request From Other Vehicle Systems, - Movement Relative to:		H01_05	ALC System Impedes Action of Other vehicle Systems	X
Uca_229	CS_149	ICL_149Request Steering Adjustment	Automated Lane Centering Control Module	Steering System	Too Long	Adjust Vehicle's Lateral Position in the δ Direction		- Direction of δ is away from the reference trajectory, δ in the opposite direction of δ, - Lateral Adjustment Request From Other Vehicle Systems, - Movement Relative to:				X

Figure 44 STPA – UCA analysis editor

Note that the analysis is usually guided by a set of guidewords. In safetbox these can be set directly in the application settings or overridden in the specific UCA editor. This allows a specific catalogue of guidewords to be set according to the system being analyzed, or according to the perspective of the analysis. E.g. Safety / Security.

Another important part of this step is to define requirements (e.g. safety / security) for the identified critical UCAs. This will occur in the requirements section of the editor, in which only the UCAs mapped to a hazard/threat/ undesired states will be shown. This view allows to see existing requirements associated to the hazard as defined in the

4 Modeling Capabilities

purpose definition step or as well as requirements associated with the control actions. Finally, the requirements specific due to the UCA can be specified.



UCA ID	UCA Summary	UCA Undesired States	Traced requirements	UCA specific requirements
UCA_205	Request Steering Adjustment (Not Provided) Source -> target: (Automated Lane Centering Control Module -> Steering System) Description: "Adjust Vehicle's Lateral Position in the δ Direction" Comments: "" Context variables: "Direction of δ is away from the reference trajectory. In the same direction as δ . Lateral Adjustment Request From Other Vehicle Systems. Movement Relative to Reference Trajectory"	(Hzd_05) ALC System Impedes Actions of Other Vehicle Systems		
UCA_210	Request Steering Adjustment (Unintended Provided) Source -> target: (Automated Lane Centering Control Module -> Steering System) Description: "Adjust Vehicle's Lateral Position in the δ Direction" Comments: "" Context variables: "Direction of δ is away from the reference trajectory. In the same direction as δ . Lateral Adjustment Request From Other Vehicle Systems. Movement Relative to Reference Trajectory"	(Hzd_02) Excessive Lateral Adjustment Resulting in Lane/Roadway Departure While ALC Is Engaged		
UCA_219	Request Steering Adjustment (Unintended Provided) Source -> target: (Automated Lane Centering Control Module -> Steering System) Description: "Adjust Vehicle's Lateral Position in the δ Direction" Comments: "" Context variables: "Direction of δ is away from the reference trajectory. In the opposite direction of δ . Lateral Adjustment Request From Other Vehicle Systems. Movement Relative to Reference Trajectory"	(Hzd_02) Excessive Lateral Adjustment Resulting in Lane/Roadway Departure While ALC Is Engaged. (Hzd_05) ALC System Impedes Actions of Other Vehicle Systems		
UCA_227	Request Steering Adjustment (Not Provided) Source -> target: (Automated Lane Centering Control Module -> Steering System) Description: "Adjust Vehicle's Lateral Position in the δ Direction" Comments: "" Context variables: "In both the opposite and the same direction as δ . Direction of δ is away from the reference trajectory. Lateral Adjustment Request From Other Vehicle Systems. Movement Relative to Reference Trajectory"	(Hzd_05) ALC System Impedes Actions of Other Vehicle Systems		
UCA_228	Request Steering Adjustment (Unintended Provided) Source -> target: (Automated Lane Centering Control Module -> Steering System) Description: "Adjust Vehicle's Lateral Position in the δ Direction" Comments: "" Context variables: "In both the opposite and the same direction as δ . Direction of δ is away from the reference trajectory. Lateral Adjustment Request From Other Vehicle Systems. Movement Relative to Reference Trajectory"	(Hzd_02) Excessive Lateral Adjustment Resulting in Lane/Roadway Departure While ALC Is Engaged. (Hzd_05) ALC System Impedes Actions of Other Vehicle Systems		
UCA_236	Request Steering Adjustment (Not Provided) Source -> target: (Automated Lane Centering Control Module -> Steering System) Description: "Adjust Vehicle's Lateral Position in the δ Direction" Comments: "" Context variables: "Direction of δ is toward the reference trajectory. Lateral Adjustment Request From Other Vehicle Systems. Movement Relative to Reference Trajectory"	(Hzd_01) Insufficient Lateral Adjustment Resulting in Lane/Roadway Departure While ALC Is Engaged		
UCA_238	Request Steering Adjustment (Too Long) Source -> target: (Automated Lane Centering Control Module -> Steering System) Description: "Adjust Vehicle's Lateral Position in the δ Direction" Comments: ""	(Hzd_02) Excessive Lateral Adjustment Resulting in Lane/Roadway Departure While ALC Is Engaged		

Figure 45 STPA – UCA analysis – Requirements view

4.4.4 Identification of Causal Scenarios

The construction of causal scenarios takes place primarily after the critical UCAs have been defined, as the aim is to identify more concrete reasons why they might occur. For this reason, the creation of a causal scenario analysis within the STPA Manager will require the selection of a UCA analysis.

The Causal Scenarios Analysis editor will allow scenarios to be specified using guide-words in a similar way to the UCA analysis. It will also display a new Control Structure Diagram representation, where the current UCAs can be easily displayed. Furthermore, this editor will directly display the requirements associated with the UCA as defined in the previous step, as well as the requirements tracing to the hazards and control actions. The user will also be able to define causal scenario specific requirements, see Figure 46.

Note that in STPA the analysis can also be performed in an inductive way, where control actions are investigated independently of a UCA. The Causal Scenario Analysis editor provides a dedicated section for this.

4 Modeling Capabilities

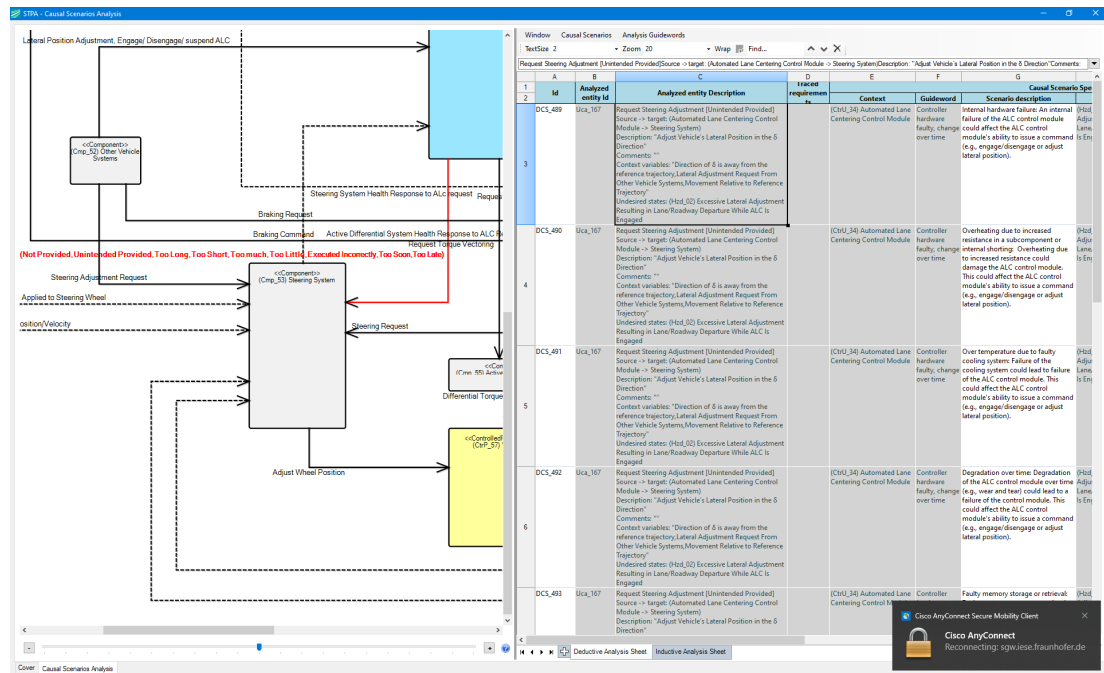


Figure 46 STPA – Causal Scenarios Analysis editor

4.5 Safety Argumentation

In this section, we describe the means the user has to create arguments about the system safety.

4.5.1 Goal Structuring Notation.

"GSN is a graphical argumentation notation that can be used to document explicitly the individual elements of any argument (claims, evidence and contextual information) and, perhaps more significantly, the relationships that exist between these elements (i.e. how claims are supported by other claims, and ultimately by evidence, and the context that is defined for the argument). Arguments documented using GSN can help provide assurance of critical properties of systems, services and organizations (such as safety or security properties)". For more information on this technique, please refer to the website: <http://www.goalstructuringnotation.info/>

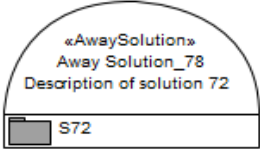
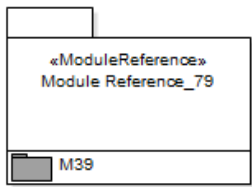
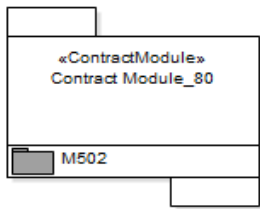
4.5.2 Goal Structuring Notation Constructs

Construct	Description	Toolbox icon	Diagram Shape

4 Modeling Capabilities

Goal	Presents a claim forming part of the Argument. A type attribute helps representing better the context of use. E.g. Safety requirement		
Strategy	Describes the nature of the inference that exists between a goal and its supporting goal(s)		
Solution	Presents a reference to an evidence item or items		
Context	Presents a contextual artefact. This can be a reference to contextual information, or a statement		
Justification	Presents a statement of rationale		
Assumption	Presents an intentionally unsubstantiated statement		
Away Goal	This repeats a claim presented in another argument module, which is used to support the argument in the local module.		
Away Context	Repeats a contextual artefact		

4 Modeling Capabilities

Away Solution	Repeats a reference to evidence items presented in another argument module		
Module Reference	Presents a reference to a module containing an argument		
Module Contract	presents a reference to a contract module containing definition of the relationships between two modules, defining how a claim in one supports the argument in the other		
Supported By	Allows inferential or evidential relationships to be documented. Inferential relationships declare that there is an inference between goals in the argument. Evidential relationships declare the link between a goal and the evidence used to substantiate it		
In context of	Declares a contextual relationship		

The following figure shows a typical example of an argument module modeled by means of GSN.

4 Modeling Capabilities

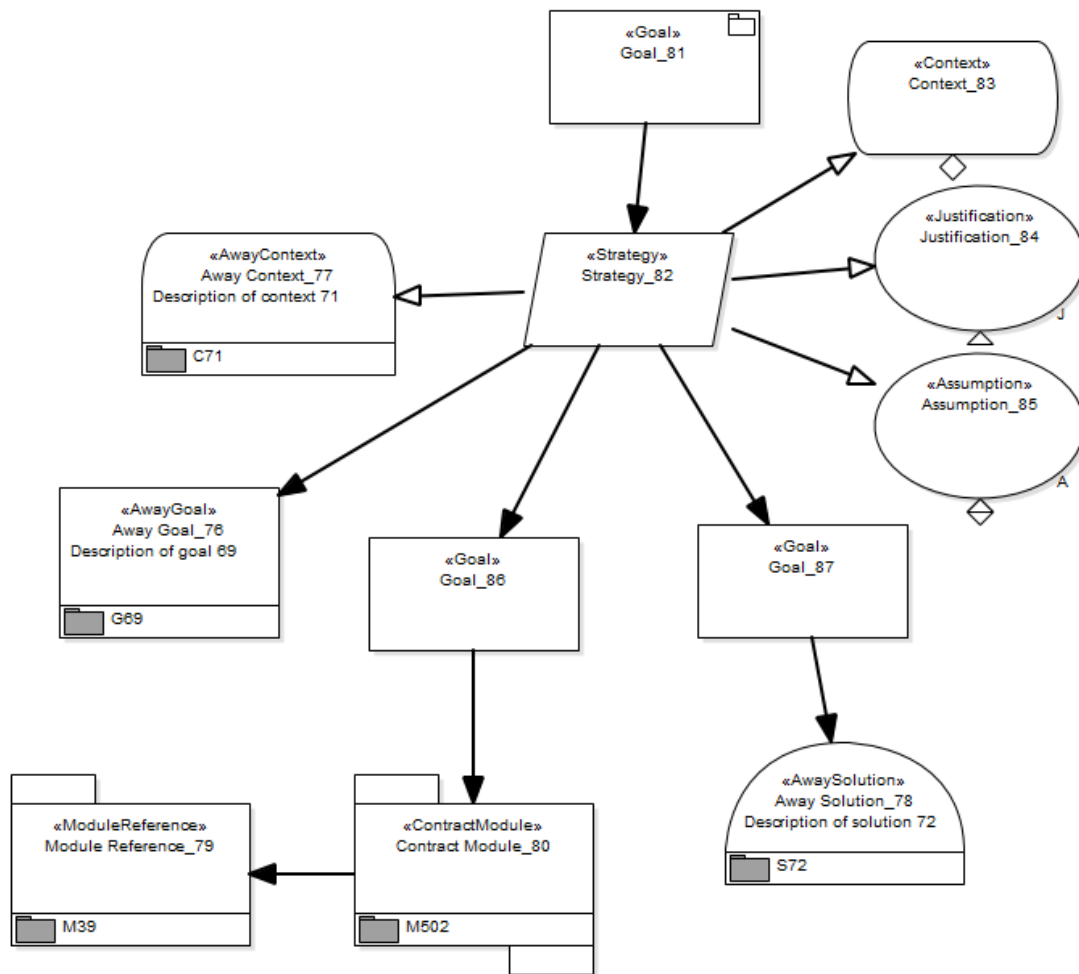


Figure 47 Argument Module (GSN) Example

5 Analysis Capabilities

In this section, you will get to know the analysis capabilities implemented in safeTbox. These can be categorized into qualitative and quantitative fault tree analyses. Before you can start executing any kind of analysis, you should configure the computation backend.

5.1 Reliability Workbench Backend Configuration

In the current implementation, two computation backends can be used for performing analyses: The internal built-in solution (implemented by Fraunhofer IESE) and the Reliability Workbench v13 incl. FaultTree+ (Isograph). The built-in backend does not require any configuration to execute the analysis, so if you want to only use this backend, you can skip this section. For the configuration of the FaultTree+ backend, the following instructions should be considered.

Note that the FaultTree+ integration feature is not active in the basic edition of safeTbox. If you are interested in using this feature, please contact us via support@safeTbox.de

In order to be able to execute the analysis with FT+, you have to first configure the license validation mechanism. FaultTree+ offers three license types that can be used: Server, Standalone, or Hardware. The required parameters are described in the following and can be entered in the safeTbox settings dialog as shown in Figure 48:

Server - A license server in your company centrally manages floating licenses; therefore, the name of the licensing host machine as well as that of the access port needs to be provided. The information <portnumber> and <hostmachinename> needs to be entered into the respective field.

Standalone - FaultTree+ standalone licenses are typically distributed as .html files, which are called EntitlementCertificate.html by default. The file location of this certificate needs to be given in the File location field. Note that the license file must be the license for the FT+ DLL itself and not the one for the Reliability Workbench tool.

Hardware - When a hardware license (dongle) is used, no configuration is needed except for the selection of the license type Hardware.

You can check your entered configurations by clicking the "Validate License" button.

Once you are done with these configurations, click on OK and the changes will be stored permanently. These application settings are automatically used for all projects.

5 Analysis Capabilities

Integrated computation and FT+ project export

The checkbox “Integrated Computation” indicates that the analysis shall be performed within FaultTree+, but the results shall be displayed within safeTbox. This is the default mode. In addition, if the analyst desires to store the Reliability Workbench file created during the Fault Tree analysis to the hard disk, the option “Export Ft+ projects to” needs to be checked. The user can then open this file with Reliability Workbench V13, for example in order to examine the created flat fault tree or to use specific RWB features on the model that are not implemented as part of safeTbox.

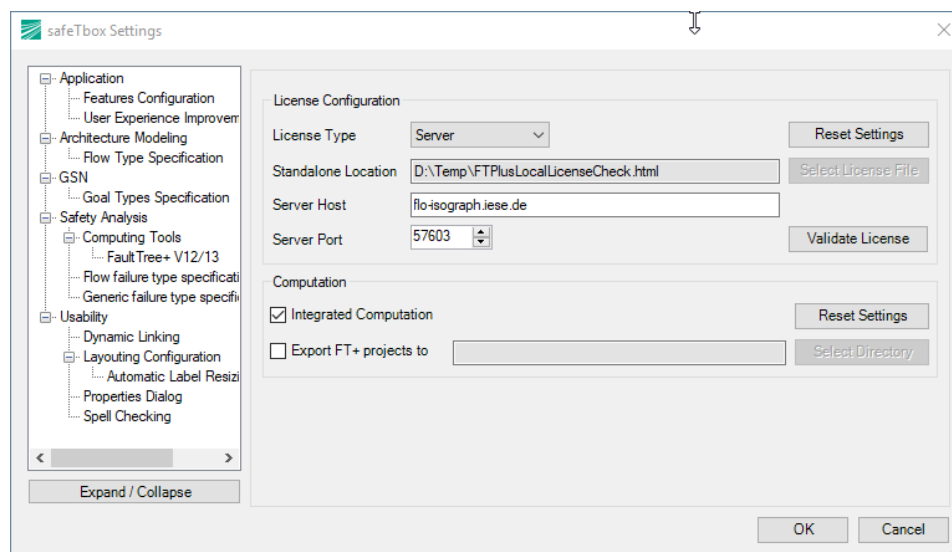


Figure 48 FaultTree+ configuration

5.2 Analysis overview

Qualitative and quantitative analyses have been implemented for (component) fault trees. Both types are performed at the same time through the same smart menu action, namely “Perform Analysis...”, see Figure 49. Moreover, both analysis types require the same input values. The action can be triggered on a selection of the following elements:

1. Output Failure Modes of CFTs
2. Output Failure Modes of CFT Instances
3. CFT Instances
4. CFTs
5. Gates in CFTs

Within the safeTbox analysis dialog, the selected computation engine (IESE or FaultTree+) and some analysis options can be set for the current analysis run (e.g. to only perform qualitative analyses with the IESE backend for performance reasons).

5 Analysis Capabilities

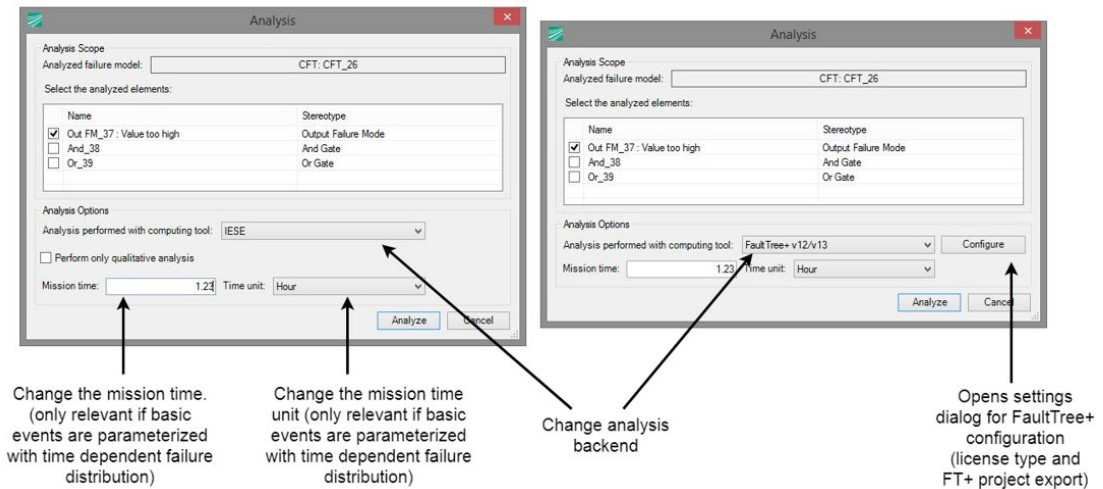


Figure 49 safeTbox analysis dialog

5.3 Analysis types

5.3.1 Fault Trees - Qualitative Analysis

The qualitative analysis that is currently available in safeTbox exists in the context of fault analysis, specifically for failure models. It is called “Prime Implicants analysis”, which is an extended version of the well-known minimal cut sets analysis for Fault Trees (it can also handle NOT logic), in which the minimal combinations of basic events will be computed for one or more output failure modes (Optionally logical gates can also be selected). For the execution of the prime implicants analysis, only the structure of the failure model is relevant., see Figure 50.

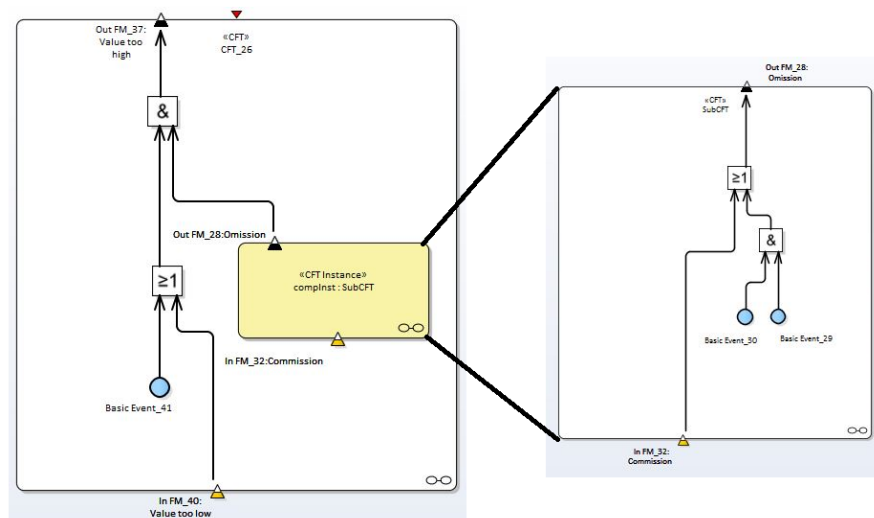


Figure 50 Simplified Component Fault Tree Example

5 Analysis Capabilities

After executing the analysis, the user will get presented the resulting combinations of events. This is presented in Figure 51. The result dialog is the same for all analysis activities and chosen computation backends (IESE or FaultTree+).

CFT Analysis Result Viewer

Element of interest: Acceleration FM : Value too high Clone dialog ☒ Show negative implicants

Occurrence probability: 3.368E-04 Mission time: 1 Hour ☐ Show Identifier

Prime implicants

#	Order	Probability	Importance	Prime implicant
1	1	4.63E-05	0.08335	Internal Failure
2	2	2.048E-04	0.3687	Emergency Shut-Off Omission, TorqueReference too high
3	2	1.482E-04	0.2668	Emergency Shut-Off Omission, RotorAngle too high
4	2	1.482E-04	0.2668	Algorithm wrongly implemented, Emergency Shut-Off Omission
5	2	2.56E-06	4.608E-03	Shut-Off Omission, TorqueReference too high
6	2	1.852E-06	3.334E-03	RotorAngle too high, Shut-Off Omission
7	2	1.852E-06	3.334E-03	Algorithm wrongly implemented, Shut-Off Omission
8	2	1.728E-06	3.111E-03	Emergency Shut-Off Omission, Internal Failure
9	2	2.16E-08	3.888E-05	Internal Failure, Shut-Off Omission

Individual implicant events Navigate

#	ID	Positive	Name	Parent path	Failure Distribution
1		True	Internal Failure	E-Drive :: PhaseCurrentSensor	Fixed (Occurrence Probability : 4.63E-05)

Event importance Navigate

#	Event name	Fussell-Vesely	Birnbaum	Parent path
1	Emergency Shut-Off Omission	0.9053	1.572	E-Drive :: Emergency Shut-off
2	TorqueReference too high	0.3733	3.24E-04	E-Drive :: AccPedalSensor
3	RotorAngle too high	0.2701	3.241E-04	E-Drive :: RotorAngleSensor
4	Algorithm wrongly implemented	0.2701	3.241E-04	E-Drive :: MicroController :: TorqueController
5	Internal Failure	0.08335	1.0	E-Drive :: PhaseCurrentSensor
6	Shut-Off Omission	0.01131	1.571	E-Drive :: MicroController :: CurrentsPlausiCheck
7	Internal Failure	3.15E-03	3.24E-04	E-Drive :: DriverController

☐ Create project backup Create result artifact(s) Export to Excel Close

Figure 51 CFT Analysis Result Viewer

In this dialog, you can see the results of the selected target elements. In the top list, you will see the MCS. In the middle list, you will be able to see the details of a specific cut set (e.g. the hierarchization path to the basic failure modes).

On the top right corner, you will find the possibility to show and hide negative implicants or events. This will certainly be the case whenever you make use of NOT or XOR gates. Next to this option, you will find the possibility to clone the dialog. This option has been introduced so that the user can compare the results of different analyzed objects at the same time.

On the bottom left corner, you will find the possibilities to export the results to Excel, as well as to store the results with the model in the form of a results artifact. During the storage process, you will be asked if you want to create a backup (checkbox on right bottom corner), see Backup Management section for more details.

5 Analysis Capabilities

Figure 52 shows the analysis results artifact that can be created results presenter. The user can create as many artifacts as desired and he will be able to load those results set afterwards at any moment.

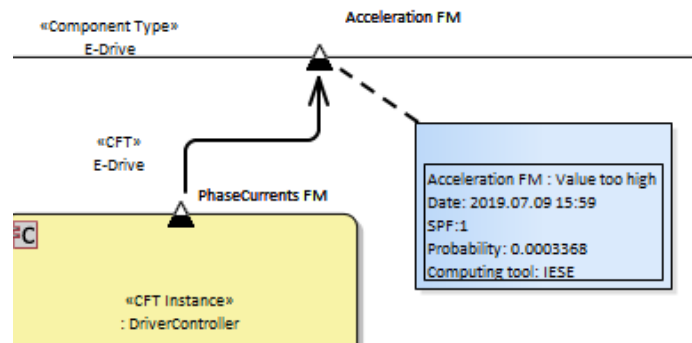


Figure 52 Analysis results artifact

5.3.2 Fault Trees - Quantitative Analysis

As mentioned above, both quantitative and qualitative analyses can be performed together when executing the analysis through the smart menu in safeTbox. Regarding quantitative analysis, it primarily computes the unavailability of the elements of interest. As a precondition for the quantitative analysis, basic events need to be parameterized with suitable failure models that express the characteristic distribution of failure occurrence probabilities for the failure modes. Currently, there are five supported failure model types, which are listed in the following table together with their required parameters. Note that the names of failure model types and their parameters are similar to the ones in FaultTree+.

Fixed	Unavailability
Rate	Failure Rate Failure Rate Unit
Dormant	Dormant Failure Rate Dormant Failure Rate Unit Test Interval Interval Time Unit
Time At Risk	Time At Risk Failure Rate Time At Risk Failure Rate Unit Time At Risk Time At Risk Unit
Uniform	Lower Bound Upper Bound

5 Analysis Capabilities

The configuration of Basic events can be performed through the properties or properties dialog, see Figure 53. The “Failure Distributions” property describes the failure model to be used for the particular element. The choice of a particular failure model indicates which of the parameters below will be used during the analysis; e.g., if we choose Fixed as failure model, the next step is to fill the parameter Unavailability.

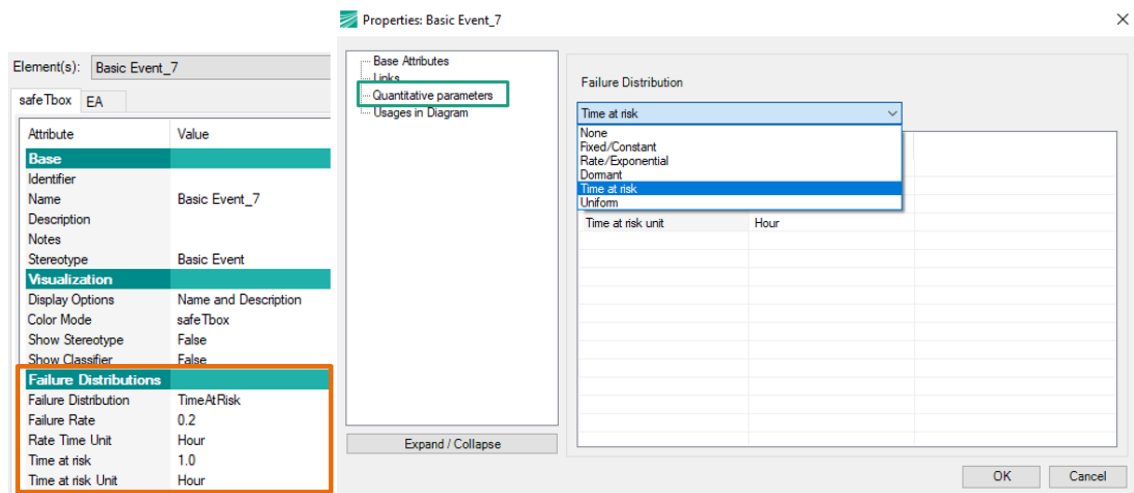


Figure 53 Quantitative parameters for basic events (left: Properties View, right: Properties Dialog)

In the case of Input Failure Modes of CFT Instances, the situation is a bit different, since not all will have to be parameterized. Input Failure Modes of CFT/C²FT instances which have incoming connections (i.e. a failure is propagated into them) do not require to be parameterized, since the propagated failure will include with it a computed occurrence probability. This is for instance the case for the Input Failure Modes Ifm_1 and Ifm_2 of the Figure 54 (left). For those Input Failure Modes, the occurrence probability is computed at the basic event “Error1” and “Incoming Error” respectively. For the same reason, it becomes necessary to parameterize the Input Failure Modes “Incoming Error”, as well as “Ifm_3”.

Another important aspect to know if the parameterization of Input Failure Modes is necessary or not, is the context of the analysis, which is determined by the topmost component for which the analysis is executed. In Figure 54, two different contexts of analysis are shown: the “System” (left) and the “CompA” (right). If the user navigates into the component instance “CompAInst” show on the left, he will end in the realization view of that component, displayed in right side of the figure. By taking this as the context of the analysis, it now becomes necessary to parametrize the Input Failure Modes “Ifm_1”, “Ifm_2” and “Ifm_3”, since they appear not to have any incoming connection from this perspective. However, if those Input Failure Modes were parameterized and the user went back to the level of the “System” component, the parameterization will have no effect and will be overwritten by the occurrence probability of the propagated failures, as explained before.

5 Analysis Capabilities

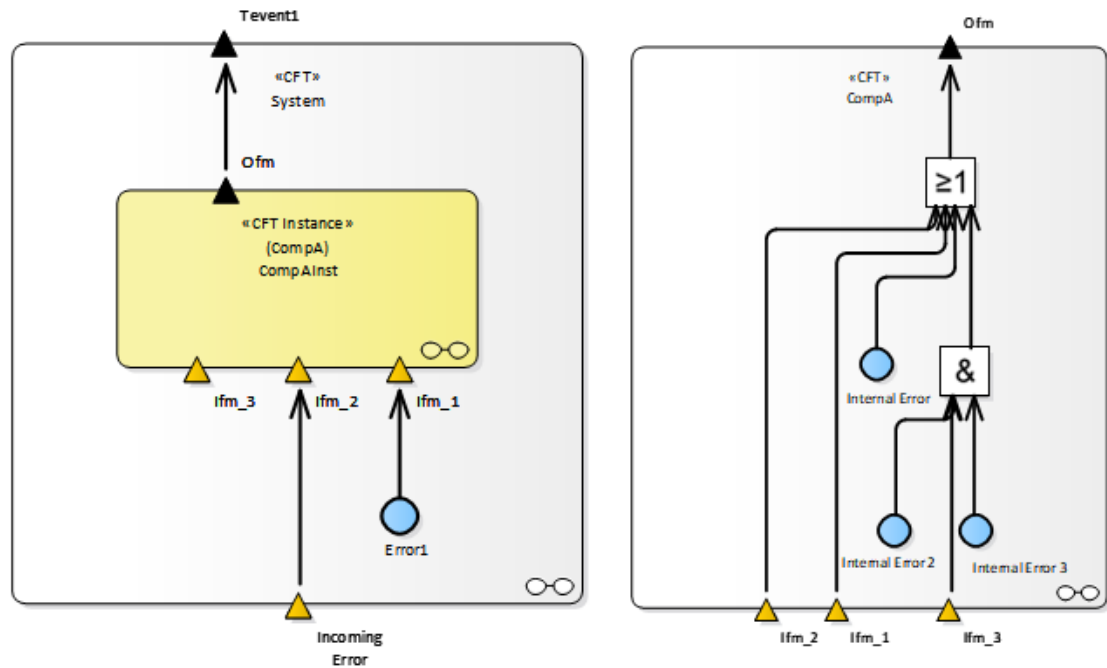


Figure 54 Failure Input Parameterization example

Once all basic events as well as all Input Failure Modes without incoming connections have been parameterized, the analysis can be triggered according to the possibilities mentioned in section 5.2. Note that prior to the actual analysis the model is checked for missing parameterizations. If these are found, a dialog (Figure 55) is shown to the user, where a temporary parameterization can be performed. This comes in handy, when quantitative data is not available yet for some basic events. In this case, a fixed probability of either 0.0 (=failure shall be ignored in the analysis) or 1.0 (=failure is always occurring) can be chosen. In any case, setting the probability in this way is only a temporary solution. This means as soon as quantitative data is available, the basic events/input failure modes shall be parameterized as shown in Figure 53.

5 Analysis Capabilities

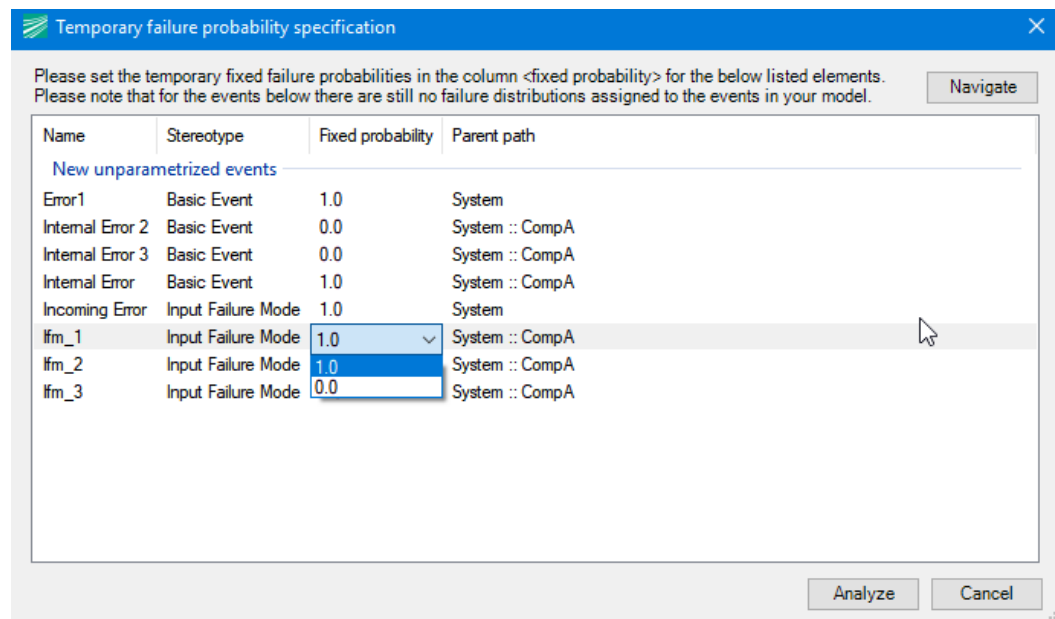


Figure 55 Temporary failure probability specification

After the analysis has been performed, the result dialog is automatically opened (see Figure 51). It contains the following quantitative computation results for the selected elements of interest:

- Output Failure Mode / Gate unavailability
- Minimal cut set unavailability
- Minimal cut set importance
- Event importance (Fussell-Vesely)
- Event importance (Birnbbaum)

The algorithms used for computing the quantitative results with the built-in backend are the same as those used by Reliability Workbench (concrete mathematical formulas for the computations are explained in detail in the regular Reliability Workbench 13.0 Help). This means, in particular, that for all calculations except Output Failure Mode / Gate unavailability, an approximation is used for top event unavailability for performance reasons. Instead of using Binary Decision Diagrams (BDD) to derive the top event unavailability, the so-called rare approximation uses the sum of minimal cut set unavailability as replacement for exact unavailability. The equivalent setting can be found in Reliability Workbench in Project Options → Calculation Tab → Importance Calculations Panel → "Use rare approximation".

5 Analysis Capabilities

5.3.3 Fault Trees - Analysis Context

In this version of safeTbox, the analysis context has been extended. Now, when starting the analysis from the Output Failure Mode of an instance, the other modeling elements (gates, events and other instances) on which the instance depends will also be added to the analysis context. In previous versions of safeTbox, the analysis context switched to the CFT realization view of the CFT Instance. This is still possible, but only once the user navigates to the associated realization view. The following figure shows an example of a partial analysis on different parts of a CFT. Here the analysis has been performed for different Output Failure Modes, yielding different results sets. This occurs, because the analysis context only considers the sub tree starting from the selected event downwards.

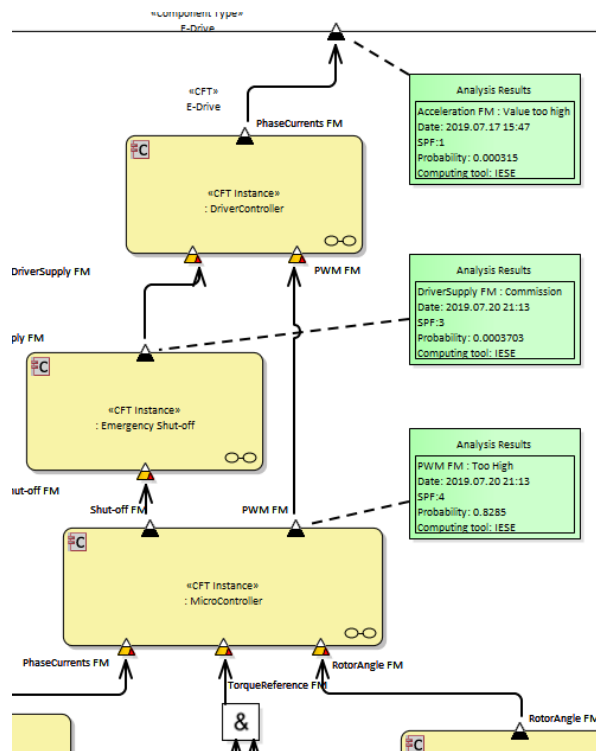


Figure 56 Analysis context example

6 Usability Features

This section describes several features that have been integrated in safeTbox for the sake of usability. The features have been conceived in such a way that they override the natural EA behavior to facilitate modeling, navigation and tool interaction, among other. These features have been modularly conceived and can be activated and deactivated by the user at any time through the “Settings management”.

6.1 Activation/Deactivation of features

You can (de-)activate individual features in the features configuration section of the safeTbox settings (Ctrl + space -> settings). As shown in Figure 57, you will see all features that have been deployed in specific version of the tool. By checking/unchecking and clicking on ok you will be activate/deactivate the feature.

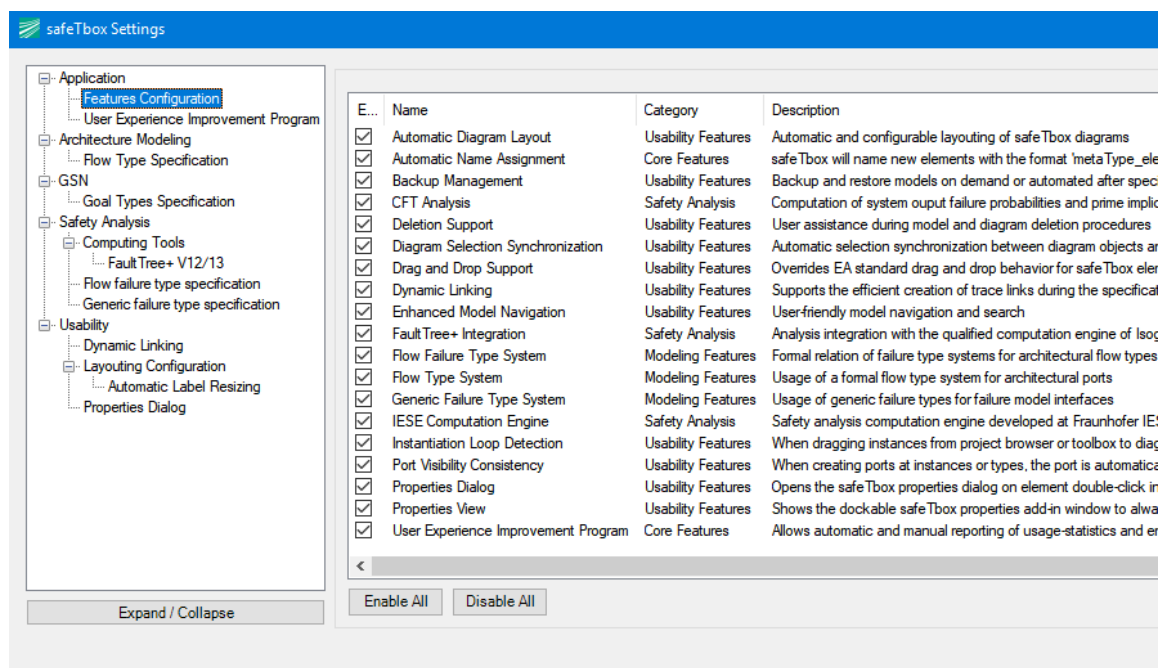


Figure 57 Features configuration

6.2 Backup Management

The Backup Management builds on top of Enterprise Architect’s *Baseline* functionality. As such, it enables the user to create snapshots of the EA project (i.e. the diagrams, models, elements, etc.), which the user will be able to restore at any time back and

7 Core Features

- **Important:** Do not delete individual EA Baselines, which are being created by the safeTbox backup functionality! Doing that will prevent the restoration of the backup containing the baseline.

6.3 Diagram and Project History

The goal of this feature is to capture a snapshot of a project's working state whenever the tool is closing, so that when the user starts working again in the same project, they will find the same working environment as the last closing time.

The working state is composed by the project browser and the list of opened diagrams. The state is stored when closing the EA instance, see next figure. This feature can be activated/deactivate through the "Settings management" -> Application -> Safe workspace state. See Figure 59.

Important: If diagrams are not available on start-up they will be skipped over.

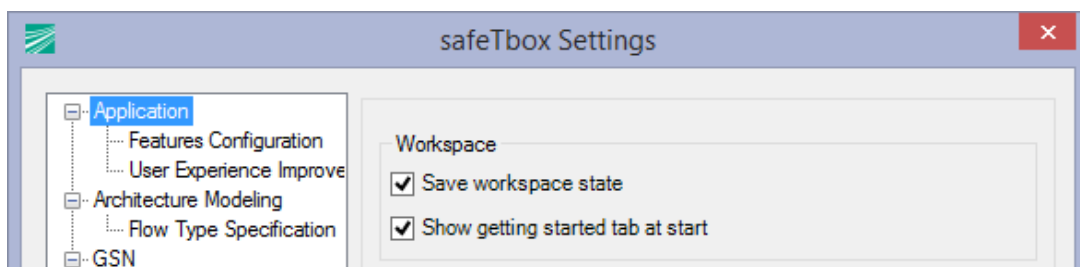


Figure 59 Application Settings

6.4 Selection Synchronization

The goal of this feature is to keep the selection of elements in a diagram consistent with the selection of elements in the project browser. The intention is to facilitate the location of elements in the project browser, without the need to do this manually (e.g. through the short-cut Alt + g). The next figure shows the configuration page of this feature. This feature can be activated/deactivate through the "Settings management" -> Application -> Synchronize diagram selections with project browser. See Figure 59.

6.5 Find and Navigation

This feature aggregates several functionalities that facilitates finding and navigating to usages or references of elements. These are:

7 Core Features

Show usages

This function will show in a dialog all the usages, links and owned elements of the selected element(s) in the diagram, grouped by the selected elements names. Note that it is only looking for this specific element, not instances/classifiers of it. A usage is depicted by the diagram in which the element appears, together with its stereotype.

In the current dialog you can navigate to usage by clicking on the "Navigate" button. According to the dialog selection, it will open the diagram and selects the usage in it.

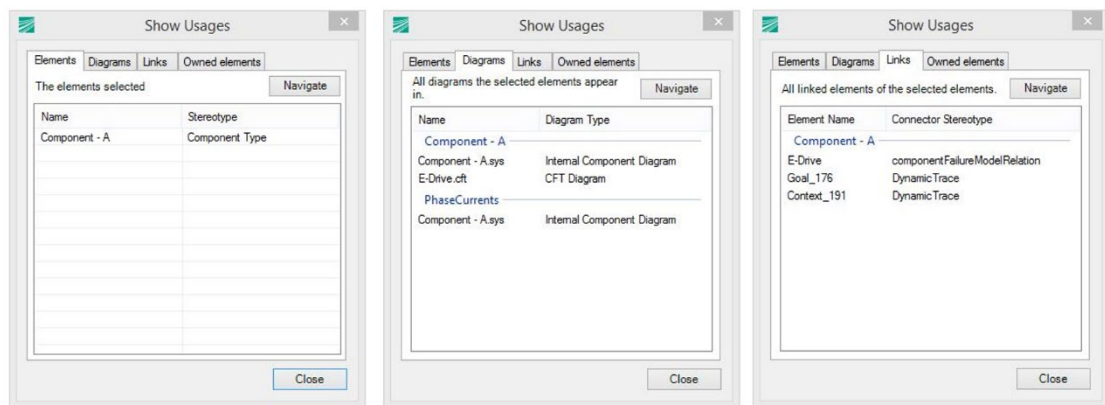


Figure 60 Show Usages dialog with single and multiple elements selected

Show Instances

This action is available on all elements that have instances of themselves in other diagrams. If an element is only instantiated once, then this action will directly navigate to the respective instance. Otherwise a dialog will open that shows the existing instances with an option to navigate to them. The options for navigation in this dialog are similar to the ones in the "Show usages" functionality. In the following figure, it can be seen than in the model the user has the possibility to navigate to one of two instances (complnst and inst2) of the same CFT, namely CFT_49.

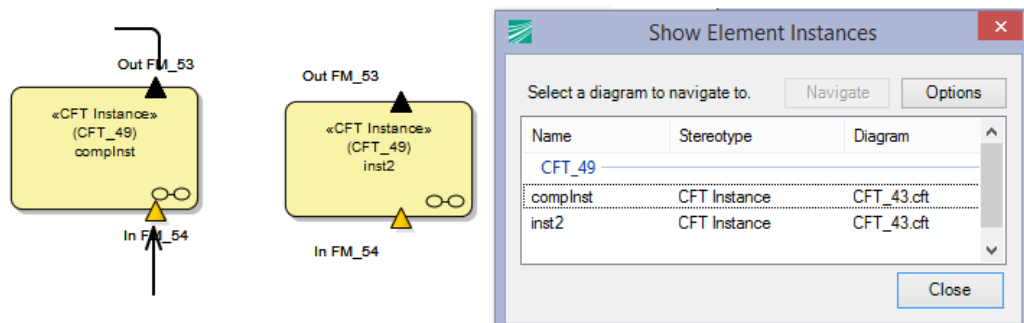


Figure 61 Navigate to Instances example

Show Realization Views

This action will show all realization views that currently exist for a Component (Instance) or a CFT (Instance). If only one realization view exists, then navigation will automatically open the respective realization view. Otherwise a dialog will open that shows the existing realization views with an option to navigate to them.

6.6 Drag and Drop

The goal of this feature is to support the user in modeling an existing safeTbox element into a diagram by making use of a drag and drop action (Dragging from the project browser and dropping in the target diagram). In safeTbox certain objects play an important role in a diagram, and the modeling by drag and drop might be restricted or supported otherwise. For example, when dragging a *Component Type* into the realization view of another *Component Type*, an instantiation procedure will be started, avoiding that the dragged component will be modeled as such in the realization.

6.7 Deletion Mechanisms

The safeTbox Deletion Mechanisms provide extended information as well as behavior when deleting elements from the project browser or diagrams. In contrast to EA's basic deletion functionality, the safeTbox deletions will also cleanup referenced elements which are depending on the deleted entity.

Example for dependent elements:

1. Deletion of a component will also delete its instances
2. Deletion of a port will also delete the port instances

safeTbox overrides the default deletion-actions provided by Enterprise Architect to protect the user from creating inconsistencies. There Smart Menu Actions available to invoke the extended deletion mechanisms. The deletion is being invoked like any other Smart Menu Action and is context-dependent:

The deletion-scenarios are:

1. Deletion from Model
 - a. Will delete elements from the project (model) and therefore it will be deleted from all diagrams
 - b. Connectors between this and other elements will also be removed from model
 - c. Irreversible
2. Deletion from Diagram
 - a. Deletes the element only from the current diagram

7 Core Features

- b. Element remains in the project (model) and in other diagrams, and it can be modeled back by drag and drop
- c. Reversible

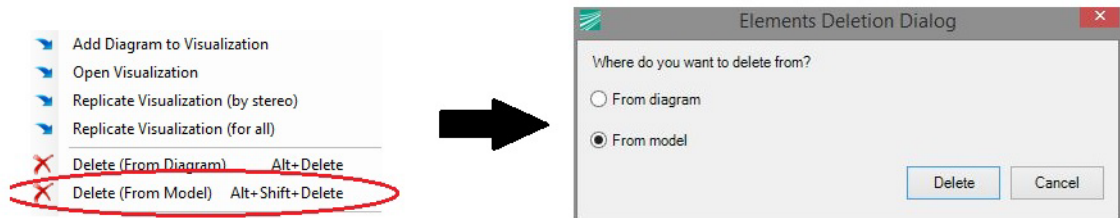


Figure 62 Deleting from model

Additional Information

- Disabling the safeTbox deletion feature will hide the Smart Menu Actions for deletion and normal EA deletions may cause inconsistencies e.g. not all instances will be deleted.
- When connectors are deleted, these are generally removed completely from model. If you want to only hide a connector within a diagram, you can do this through the Properties Dialog or “Show linked elements” dialog. Alternatively, through the EA context menu -> Visibility -> Hide Connector. See Figure 63.

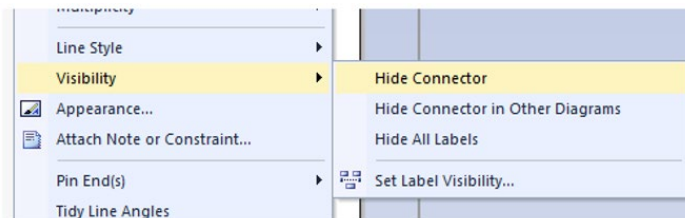


Figure 63 Hiding connectors

6.8 Port Visibility Consistency

This feature ensures that ports of component instances or CFT instances are synchronized with their preferred realization view and vice versa. If the feature is enabled and new ports are modeled on a realization view of a component, the port will be created for all instances of this component. However, it will only be visualized for those instances, which have the same realization view.

If the feature is disabled and ports are deleted from diagram, no synchronization will be performed on other instances or realization views. Thus, the diagram appearance of the models can be in an inconsistent state.

6.9 Properties Dialog

safeTbox provides its own properties dialog which resembles the properties dialog of EA. It includes several usability and traceability features to facilitate modeling activities. Furthermore, according to the object under configuration, some extra safeTbox attributes might be shown, see Figure 67.

Properties dialog can be opened by:

1. Open smart menu for an element and select the menu entry safeTbox Properties

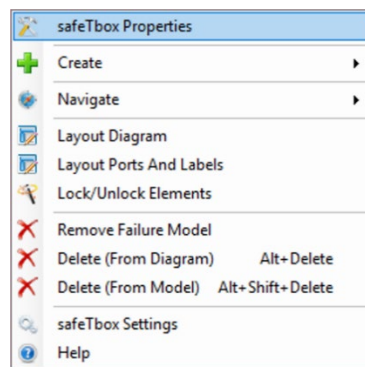


Figure 64 Calling the properties dialog through the smart menu

2. Perform a double click on the element (in diagram or project browser).

The standard properties dialog provided by EA for elements and connectors still can be accessed, through the EA context menu. However, we do not encourage its use for safeTbox supported modeling elements. This dialog will be by default used for all elements and connectors within a project. Nevertheless, it can be configured through the "Settings management" for which types and stereotypes the EA default dialog shall be preferred.

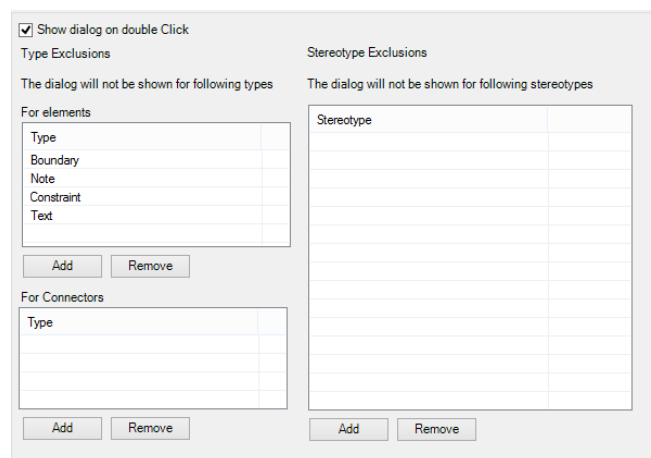


Figure 65 Properties dialog configuration page

7 Core Features

This feature can be activated or deactivated through the settings dialog, as usual. Since the Dynamic Linking feature makes part of the Properties Dialog, if you disable the Properties Dialog feature, then automatically the Dynamic trace building feature will be disable too. Moreover, there will be no smart menu entry for the safeTbox Properties anymore and double clicking on the element will open the EA default properties dialog.

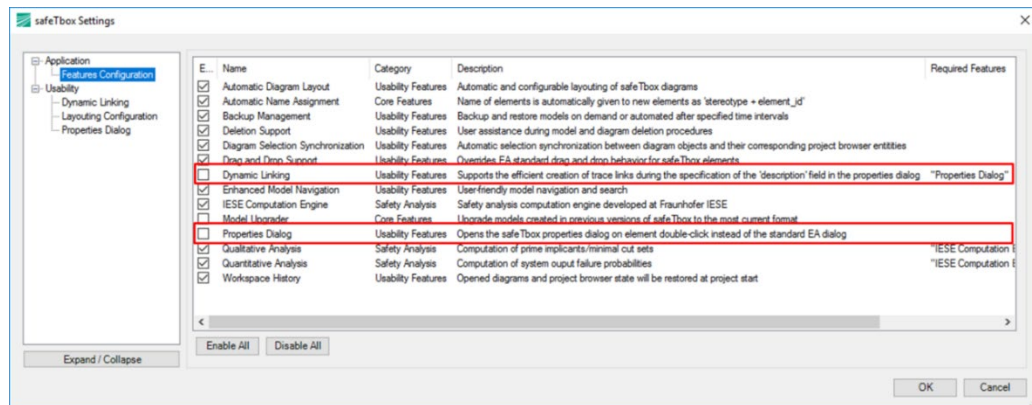


Figure 66 Enabling/disabling the Properties dialog

The field description is of special importance since it supports the “Dynamic Trace Building” feature, which will allow you to add links to other elements into the Description text box by using an autocomplete list. If this field is not part of the attributes of the element under configuration (e.g. UML Class), then the user might get asked if it needs to be added, depending whether the field has been filled or not.

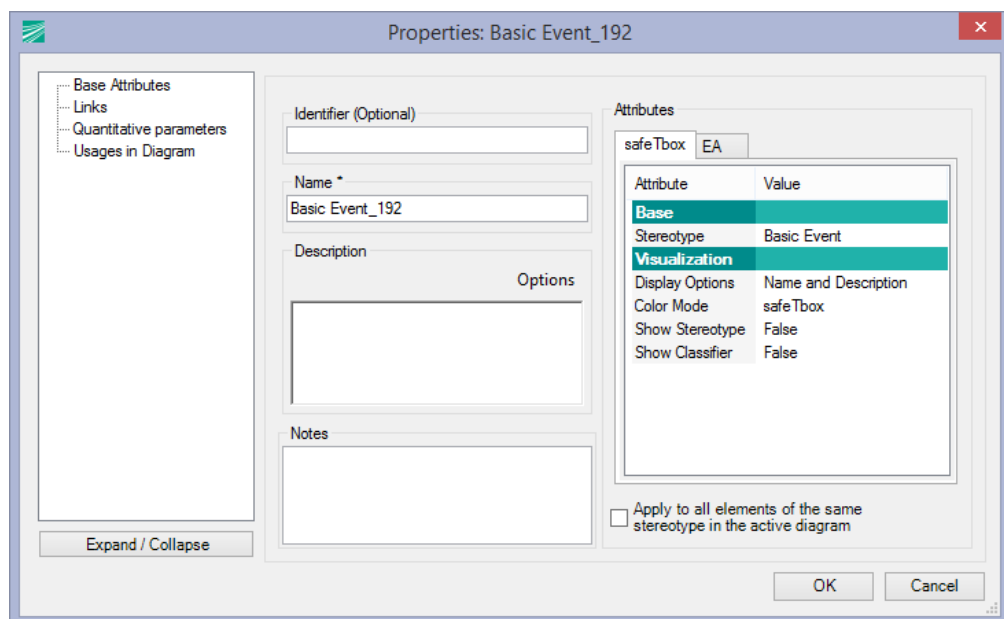


Figure 67 safeTbox properties dialog for elements

6.10 Properties View

The properties view is a user interface for displaying and modifying model-related data. It builds on top of the user experience of Enterprise Architect's native *properties* window and adds additional usability functions to it. This view is embedded into the EA Add-In window GUIs.

By using the properties view, *Tagged Values* and other entity-related (e.g. the name or stereotype) data can be displayed and modified. While the EA properties view focuses on one specific entity, the properties view supports multi-selection of multiple entities at the same time.

The EA *Add-In Window* can be enabled via EA's menu bar by selecting "Extensions → Add-In Windows". See Figure 68.

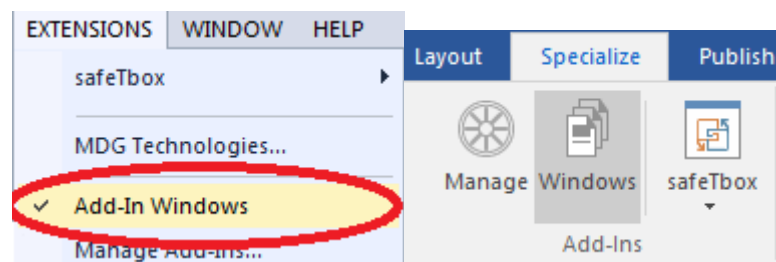


Figure 68 Enabling EA's Add-In Window (Left EA13, Right EA14)

The Properties View user-interface

The properties view consists of four major components (see Figure 69):

1. The configuration bar, which allows you to enable the loading of common attributes when performing a multi-selection, and the enabling of the selection synchronization.

Note: the synchronization might affect the modeling performance.

2. The configuration menu, which contains the menu-bar as well as additional configuration options.
3. The context selection dropdown-list shows the currently loaded entity or set of entities. In case multiple entities have been selected, the dropdown-list can be used to select either one of the entities, or all of them together for modifying their shared attributes.
4. The attributes view shows the attributes of the selected entity/entities. The tabs categorize the different attributes of the entity according to semantics. In this figure the tab "safeTbox" is selected, which shows the safeTbox-relevant attributes.

7 Core Features

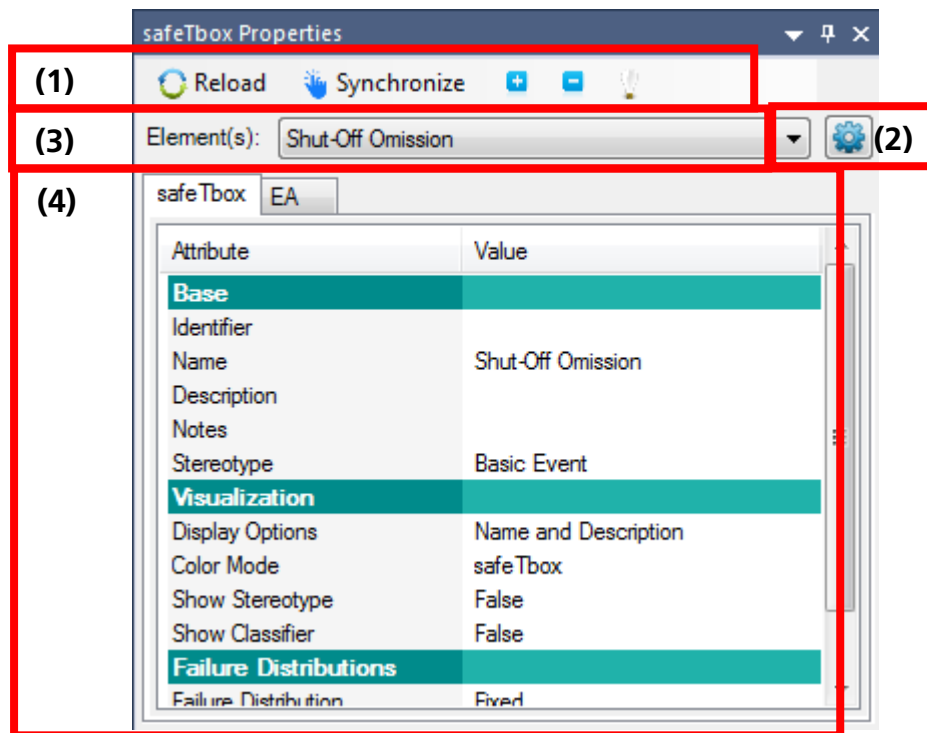


Figure 69 Properties View

Configuration Bar & Menu

The following configuration options exist within the configuration bar (in button order):

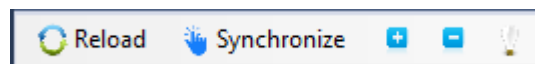


Figure 70 Properties View configuration bar

1. **Reload** will show or refresh the attributes of the EA entities which are currently selected either in the current diagram or in the project browser. Refreshing of attributes can be used in case the entities have been modified externally (e.g. by using the EA properties dialog), or when a multi-selection has been performed.
2. **Synchronize** toggles the option for automatically displaying the currently selected EA entity/entities attributes within the properties view.
3. **Show Description** toggles the visibility of a short attribute-description text at the bottom of the properties view.

6.11 Dynamic Trace Building

This feature is integrated as part of the field description in the Properties Dialog. It allows the user to establish traces to elements within the current project in a more natu-

7 Core Features

ral way. See Figure 71. When typing the user gets matching element names offered for linkage. Once one of them is selected in the autocomplete menu, automatically a trace to this element will be created. It should be noticed that this feature cannot be enabled for any kind of connectors, since EA does not support having a connector as a source or target of other connectors and will therefore not be activated for the properties dialog of the connectors.

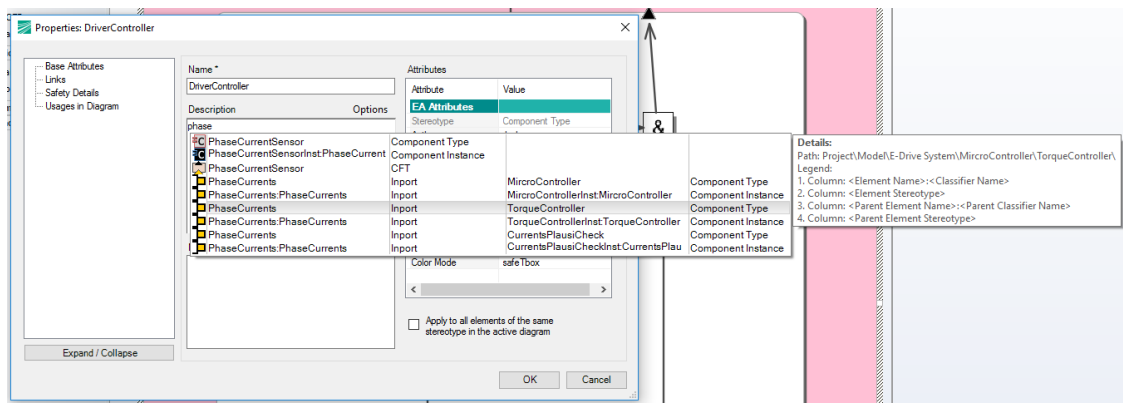


Figure 71 Dynamic Trace Building - Autocomplete example

After you had chosen an element from the list by clicking twice on it or using the Enter key, a link will be inserted into the description text box. Links have another font style compared to regular text, {Underlined, Blue font color, Begins and ends with a pipe character, is clickable}.

If you click on a link, a context menu will open. You have the ability to navigate to the linked element or remove the link from the description. If the element exists in more than one diagram, you can choose which one you want to navigate to. Moreover, if the link is broken, i.e. the trace has been removed by other means, or the associated element has been removed from model, then the user will have the possibility to repair the link or remove it respectively.

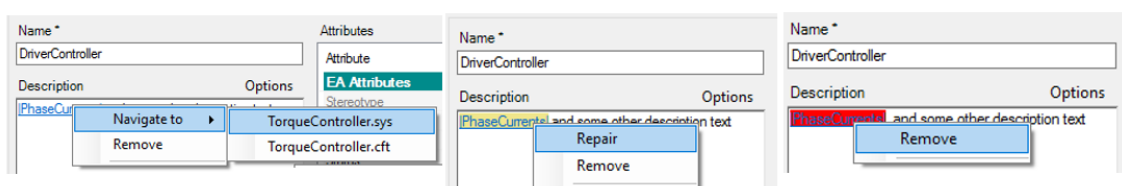


Figure 72 Dynamic Traces context menu

The creation of dynamic traces between elements has also a visual effect on the shape of the safeTbox elements. If you decide to show the description field then the created links will be shown in the text of the description field as embedded into parenthesis {"some link"}, as can be seen in Figure 73.

7 Core Features

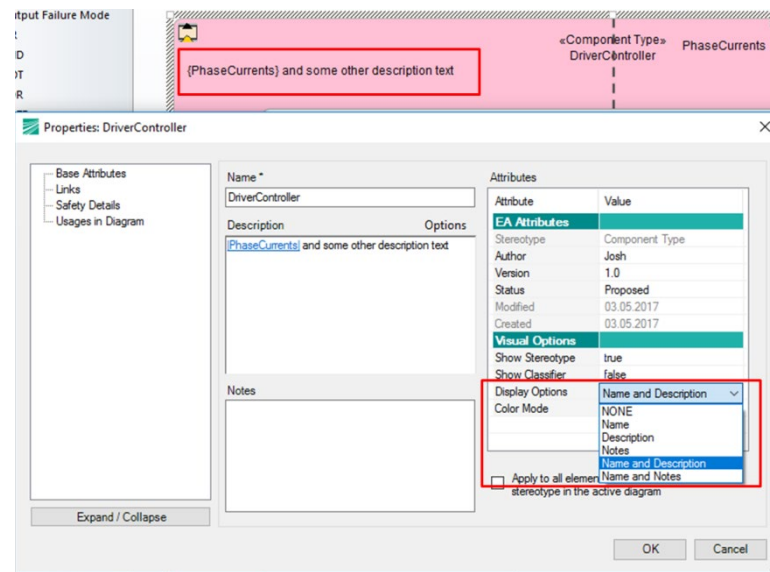


Figure 73 Dynamic Trace text on shape script

The stereotypes of the elements offered for linkage through the autocomplete menu can be configured through the settings management or directly in the properties dialog by clicking on the option button -> Set stereotype filters. See Figure 74 and Figure 75, respectively. By default, all types and stereotypes will be enabled.

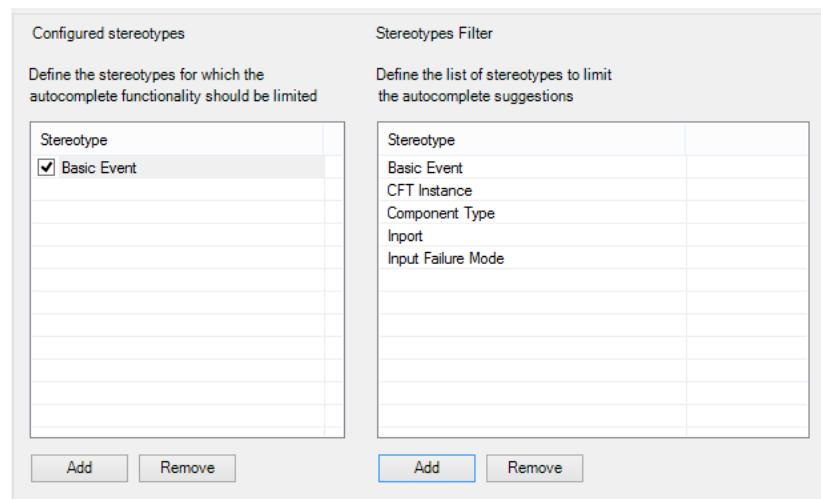


Figure 74 Configuring Dynamic Trace Building at settings management

7 Core Features

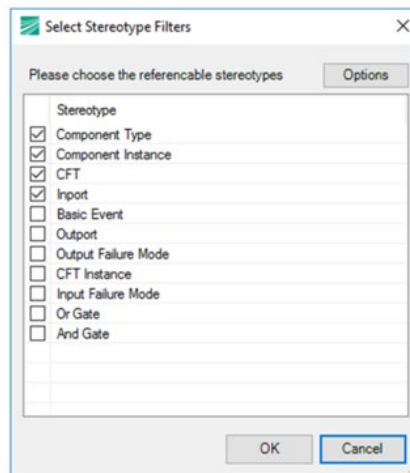


Figure 75 Dynamic Trace Building – Set filters example

When the Select Stereotype Filters dialog is opened the first time, it shows all the stereotypes, which have been configured in the Dynamic Linking settings, as checked. The settings you change here will not override the Dynamic Linking settings in the safeTbox Settings dialog. It will only allow you to change the filter for the auto completion list until the Properties Dialog is closed again.

Finally, you can enable or disable the entire feature under the settings management -> features configuration.

6.12 Diagram Layouting

safeTbox integrates its own layouting library, which allows to automatically rearrange elements and connectors for specific diagram types. While EA offers some generic layouting algorithms, in safeTbox the user will have the possibility to define several parameters relevant for positioning the elements, e.g. orientation of elements and spacing between elements. The layouting configuration is highly customizable and new configurations can be added. Currently, layouting configurations are available for safeTbox Component Type diagrams and Component Fault Tree (CFT) diagrams.

The following list shows the available layout actions that can be performed:

1. Layout Diagram

- a. This functionality can be accessed through the smart menu and can be used on the realization view of Component Type diagrams and CFT diagrams.

7 Core Features

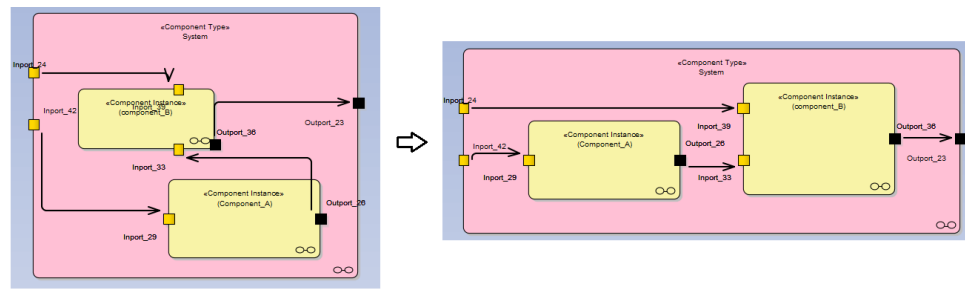


Figure 76 Layout Diagram example

2. Layout Port and Labels / Layout Labels

- This functionality can be accessed through the smart menu and can be used on diagram elements that have ports. The ports will be placed on the edges of the container element according to layouting configuration.
- Important:**
 - Ports will be rearranged proportionally to length on the edge
 - The display order will not be change after rearrangement (so connectors don't cross each other)

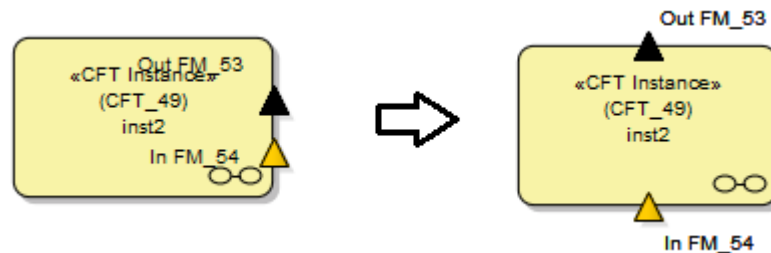


Figure 77 Layout Ports and Labels example

Layouting Configuration

The layouting configuration is available in the settings of safeTbox under Layouting Configuration. The main settings page of the layouting configuration is shown in Figure 78. There are two default layout configurations available for each diagram type and they cannot be modified. The Standard layout configuration will be sufficient for most situations and the Compact configuration can be used if the diagram is very large and should be as compact as possible. For special use cases custom layout configurations can be defined that are based on existing configurations.

7 Core Features

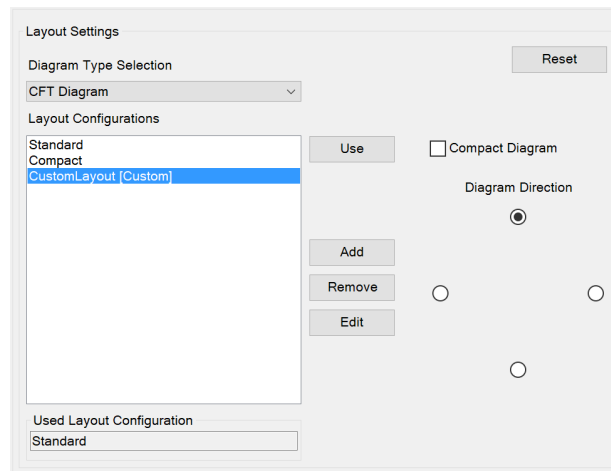


Figure 78 Layout configuration

The initial layout configuration for all diagram types is the Standard configuration. To change the used layout configuration a different one has to be selected in the list and the Use button has to be pressed. Below the list, the currently used layout configuration is visible. Be aware that the layout configuration must be set for each diagram type, meaning that if all diagrams should be layouted with the Compact configuration, the used Layout configuration must be changed for each diagram type. The 2 available default configurations cannot be edited or removed but new editable layout configurations can be added.

Custom layout configurations the diagram direction can be modified which will change how elements are placed and how connectors flow through the diagram. See Figure 79.

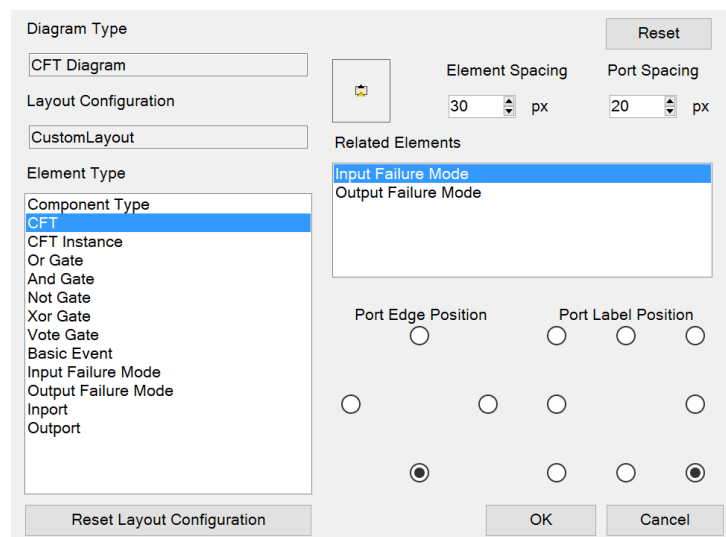


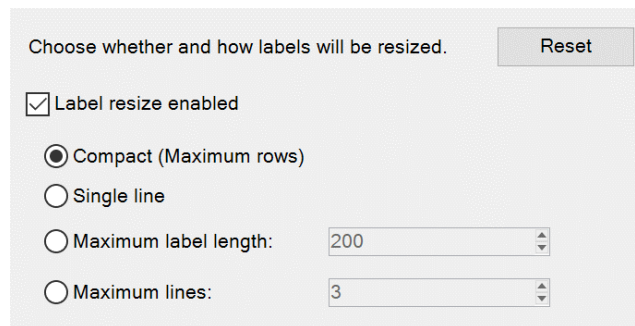
Figure 79 Advanced Layout Configuration

7 Core Features

- **Element Spacing:** The space that should be kept between this element and other elements.
- **Port Spacing:** The space that should be kept between ports of an element.
- **Port Edge Position:** On which side of the port container the port should be located.
- **Port Label Position:** Where the label of the port should be placed.

Automatic Label Resize

There is another settings page that allows to change the way labels are resized. Figure 80 shows this settings page, which is located below the layouting configuration.



The screenshot shows a settings dialog titled "Choose whether and how labels will be resized." with a "Reset" button in the top right corner. The dialog contains the following options:

- ☒ Label resize enabled
- ☒ Compact (Maximum rows)
- ☐ Single line
- ☐ Maximum label length: 200 (with a spin button)
- ☐ Maximum lines: 3 (with a spin button)

Figure 80 Automatic Label Resize

6.13 Automatic Project Upgrade and Downgrade

This feature allows the user to upgrade a project created with an older safeTbox version to the one he is using and an export a project to an earlier version of safeTbox than the one installed.

Project Upgrade

An upgrade potentially includes the renaming of element/connector stereotypes and the addition/modification of their tagged values. The upgrade is essential in order to turn the model compatible with the current safeTbox installation. If not done, new features might not work properly. Note that the upgrade procedure follows a non-deletion policy, i.e. during an upgrade, no previously modeled information will be deleted.

The safeTbox Project Version Change Assistant opens as soon as it detects that a project is outdated and therefore not compatible with the current safeTbox installation. A wizard guides the user through the upgrade. During the upgrade process a backup of the project will be created. This one can be found in the project directory under the name <yourprojectname>_Backup.eap.

Project Downgrade

This functionality will allow the user to downgrade the current opened project to a version compatible with earlier versions of safeTbox. This functionality can be accessed under Menu – Extensions – safeTbox - Downgrade Project.

When executed, a dialog showing the list of possible versions to be downgraded will show up (See Figure 81).

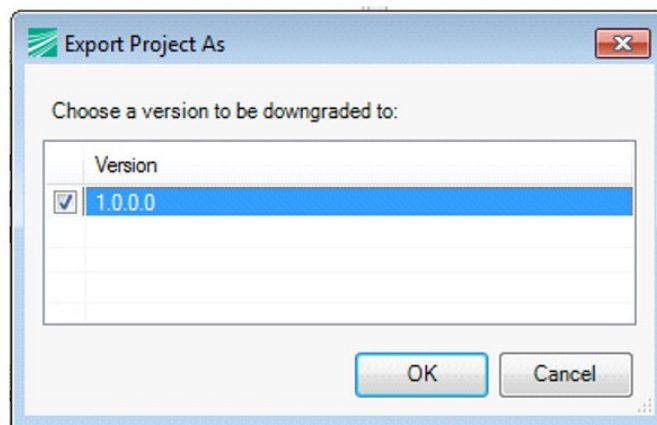


Figure 81 Export Project window

The currently opened project gets downgraded and the original project can be found as a copy in the same directory under the name <yourprojectname>_Original.eap.

6.14 Spell check

This feature supports the user at text typing in the following element and connector attributes:

- Identifier
- Name
- Notes
- Description

It offers spellchecking at the time of typing as well as suggestions for incorrectly spelled words according to the specified language. Furthermore, it allows custom words to be excluded from spell checking.

Settings configuration options

The feature can be configured through the dedicated Settings page found under Settings -> Usability -> Spell Checking

7 Core Features

Possible configurations include (See Figure 82):

- Selecting an active dictionary
- Making a dictionary default
- Adding custom words to the exclusions list
- Removing words from the exclusions list

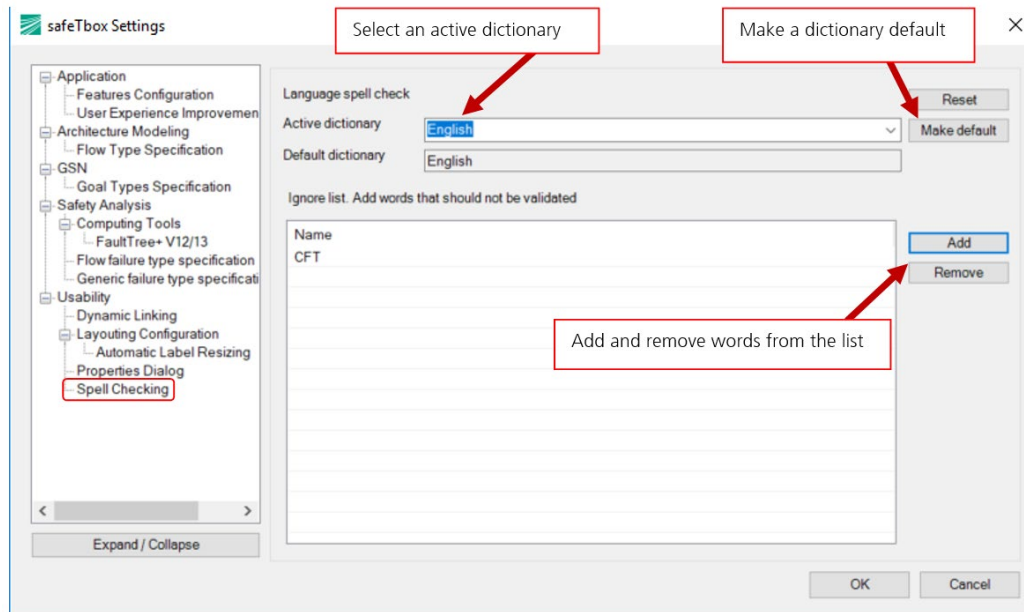


Figure 82 Spell checker Settings Page

Actions

The feature includes these actions while typing (See Figure 83):

- Highlight wrong words
- Suggest corrections
- Select a highlighted word to be ignored and put into the exclusions list

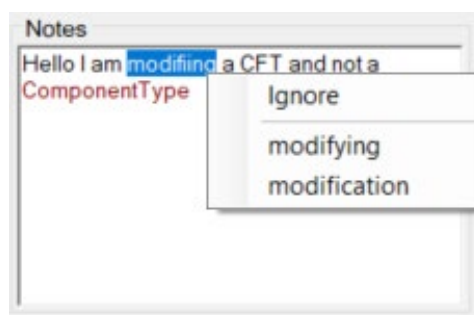


Figure 83 Ignore word

6.15 Default Visualization

This feature allows the user to set a default visualization configuration for a specific stereotype or to replicate the visualization for all the elements in the diagram. The visualization configuration includes the appearance of the element as well as the safeT-box visualization properties.

The feature includes the following actions, available in the smart menu:

- Use as default appearance
- Replicate Visualization

Use as default appearance

This action allows the user to set a default visualization configuration for a specific stereotype. All newly instantiated element of that stereotype will have the chosen visualization configuration.

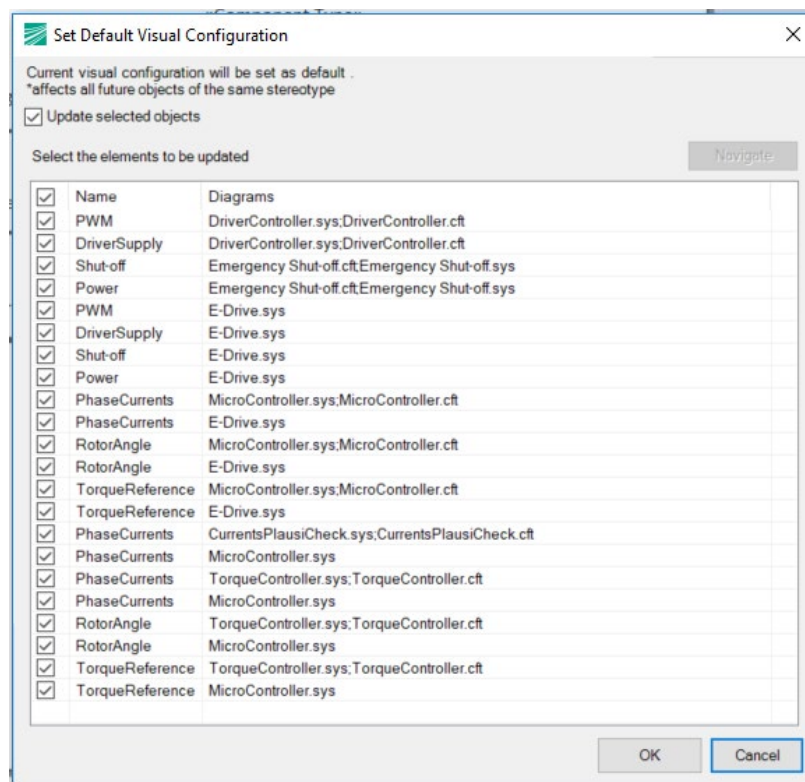


Figure 84 Set as default visualization

7 Core Features

Replicate Visualization

This action allows the user to replicate the visualization configuration of the selected element to all the elements in the current diagram.

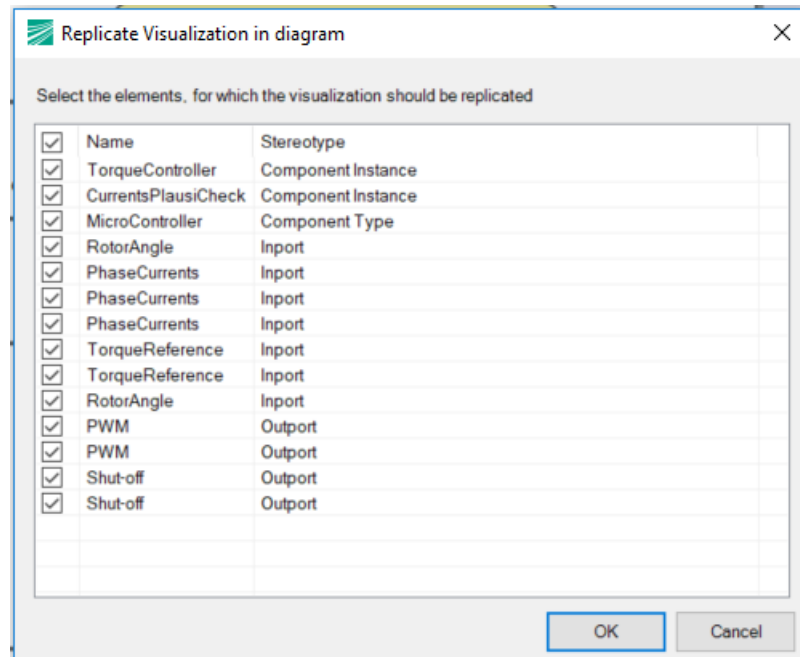


Figure 85 Replicate Visualization

6.16 Excel Exporter

This feature allows the user to export elements from EA to an excel sheet. The action "Export" in the smart menu can be used on either a single element or a single package and it will open the exporter dialog shown in Figure 86. The dialog considers the hierarchy of the selected element or package and will show all child elements that can be exported. In Figure 86 the root package "Model" is selected which means all elements under "Model" are considered. This is called the scope of the export dialog. The scope can be changed by clicking the "Select" button on the left side above the "Navigate" button. Then the project browser selection shown in Figure 87 is opened which allows to set a new scope for the export dialog. Both export dialog and project browser selection dialog allow filtering as can be seen in the figures.

To export all elements, independent of the current selection, use the "Export (All)" button. If only some elements shall be exported then they must be checked and then the "Export (Selected)" shall be clicked. In the exported excel file the elements are grouped by their stereotype. For each stereotype there is a new sheet in the excel document and the table in the sheet is different depending on the tagged values of the particular stereotype.

7 Core Features

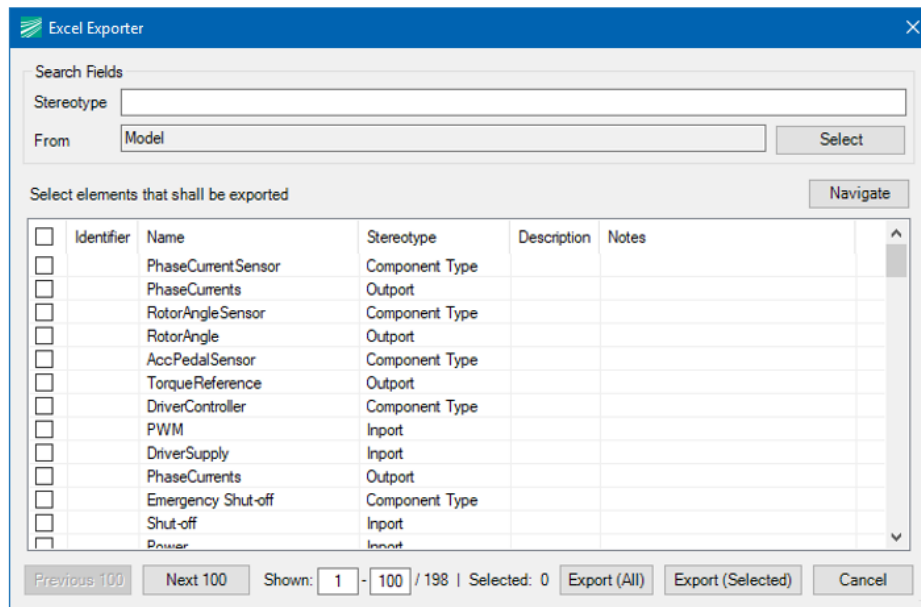


Figure 86 Export Dialog

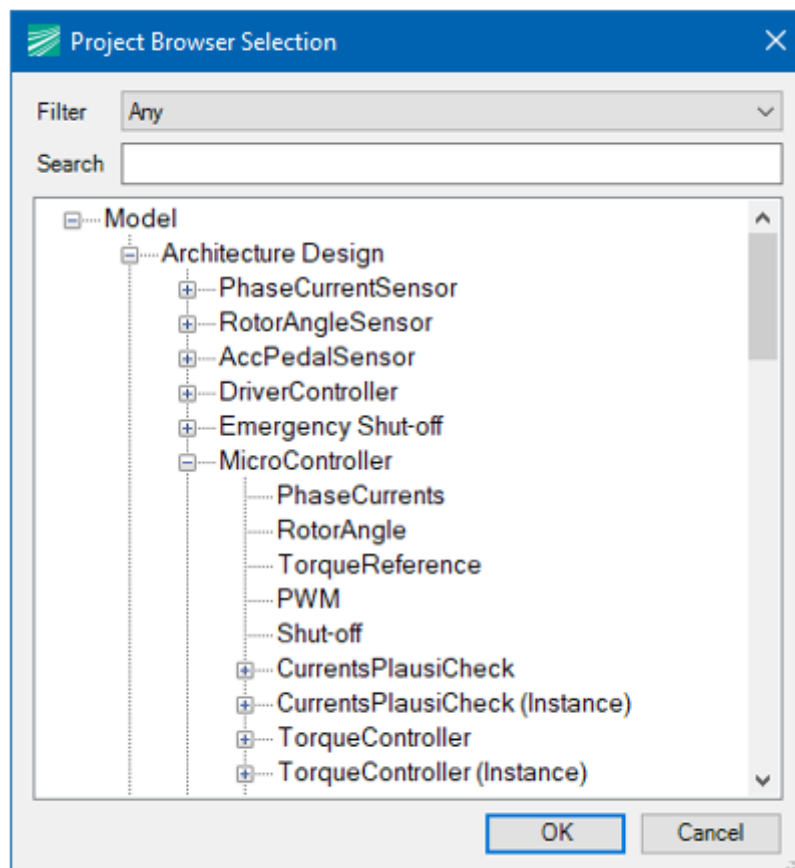


Figure 87 Project Browser Selection

7 Core Features

7.1 User Experience Improvement Program

In order to make safeTbox a better tool, this and future versions of the tool include mechanisms to log and send information to the development team. This information contains primary usage statistics, which do not relate to any model specific information, as well as debugging information, which will be sent together with bug reports in case they occur. The user has the possibility to avoid logging as well as reporting if desired through the safeTbox settings.

The user can enable or disable both at once by disabling the entire feature: Settings management -> Features configuration -> Feedback group.

Or individually (i.e. User experience reporting, Error Reporting) at Settings Management -> feedback group.

7.1.1 User Experience Reporter

The User Experience Reporter **automatically** captures and reports usage patterns that are then used by the developers at Fraunhofer to improve primary usability and performance aspects of safeTbox.

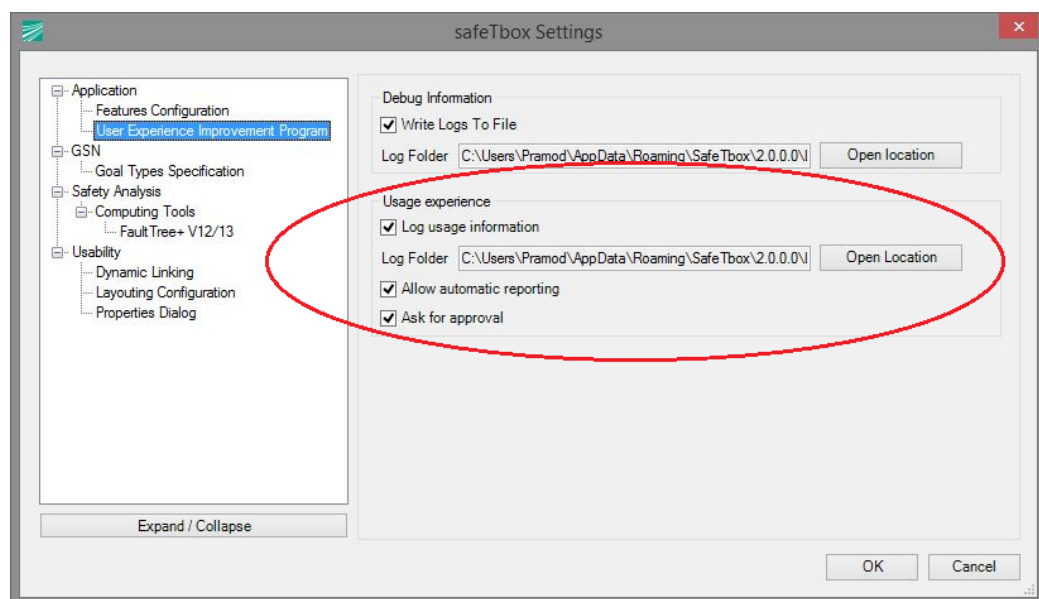


Figure 88 Usage Experience settings

7 Core Features

The user can completely disable the reporter by turning off the "Log usage information" option. Further, the user can choose to disable automatic reporting by turning off "Allow automatic reporting". By turning off automatic reporting, the usage statistics are still logged but the logs will not be reported to Fraunhofer.

If automatic reporting is enabled, the reporter attempts to send the logs 7 days after the last reporting process was successful. The user can choose to view the reports that will be sent by enabling the "Ask for approval" option, which displays a prompt before each attempt.

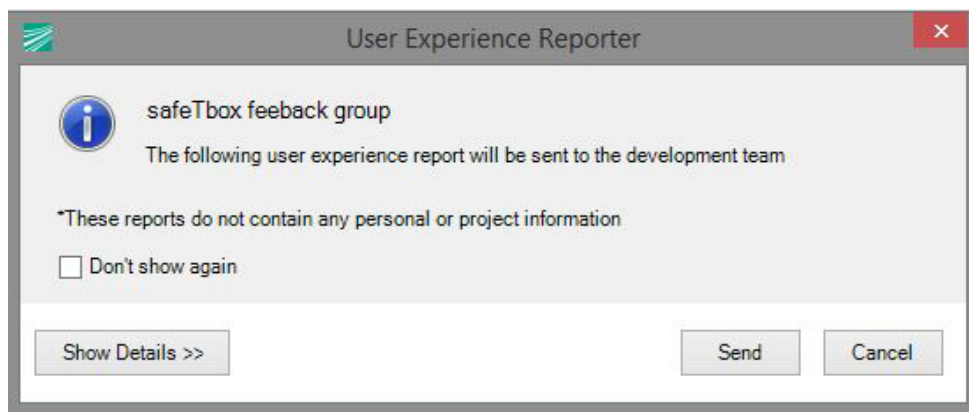


Figure 89 Usage experience – Ask for approval option

To ensure transparency, the reporter displays the information that has been prepared to be sent. This can be viewed by choosing the "Show Details" button on the prompt. A more user-friendly version of these reports will be displayed.

The prompts can be disabled by using the "Don't show again" option. This option turns off the prompt but continues reporting without interrupting the user.

7.1.2 Error Reporter

The Error Reporter creates detailed reports of the errors occurred during the operation of safeTbox. These reports give the developers at Fraunhofer important insights into any bugs that are present in safeTbox.

7 Core Features

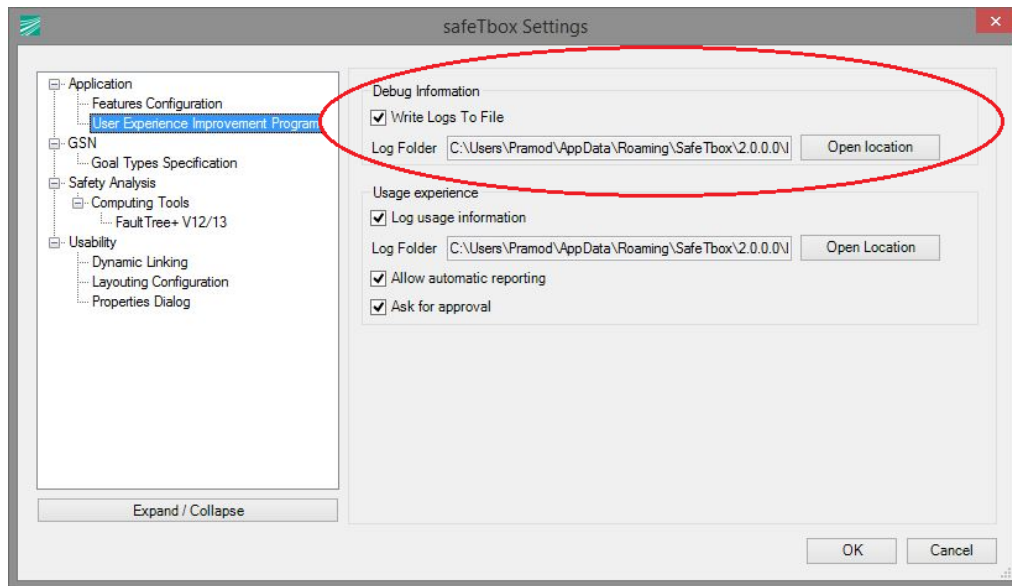


Figure 90 Debug logs settings

By choosing "Write Logs to File", you can allow safeTbox to write the debug logs in a text file and store it in the system hard drive. All the above-mentioned categories of information are stored in these log files. The location where these files are stored can be accessed by choosing the "Open location" option. When the user decides to disable this option, safeTbox stops writing debug information into files.

safeTbox automatically deletes the log files stored in the disk periodically. This will occur for files older than 7 days from the moment of their creation. These files are deleted because they are considered not useful anymore, and to avoid disk space over usage.

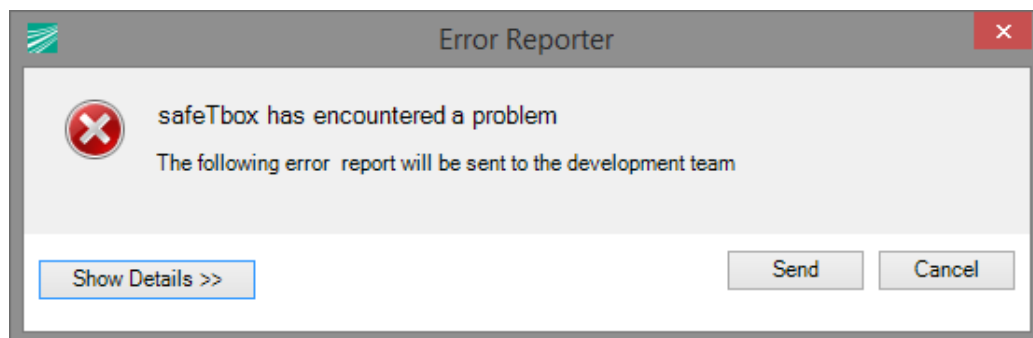


Figure 91 Error Reporter - Reporting Window

When safeTbox encounters an error, an Error Reporter window pops up. This reporter collects all the helpful debug information, and when the user clicks on "Send", it reports the collected information to the developers at Fraunhofer. The "Show Details" option provides a detailed log of the error that occurred and its stack trace and also a list of events that led to this error.

8 Annexes

8.1 Annex 1

1. S. Velasco, Towards proper tool support for component-oriented and model-based development of safety critical systems, Proceedings of the 4th Commercial Vehicle Technology Symposium, Kaiserslautern, Germany, CVT 2016.
2. M. Kaessmeyer, S. Velasco, M. Schurius, Evaluation of a systematic approach in variant management for safety-critical systems development, Proceedings of the International Conference on Embedded and Ubiquitous Computing, Porto, Portugal, EUC 2015.
3. P. Antonino, S. Velasco, M. Trapp M., J. Reich. iSaFe: An integrated Safety Engineering Tool-Framework. Proceedings of the International Workshop on Dependable Control of Discrete Systems, Cancun, Mexico, 2015.
4. J. Schwinn, R. Adler and S. Kemmann Combining Safety Engineering and Product Line Engineering. In Software Engineering 2013 Workshop, Aachen, Germany, 2013.
5. Adler, R. et al.: "Integration of Component Fault Trees into the UML", proceedings in Models in Software Engineering, Workshops and Symposia at MODELS 2010 - Reports and Revised Selected Papers, Pages 312-327 Springer-Verlag Berlin, Heidelberg 2011
6. D. Domis and M. Trapp Integrating Safety Analyses and Component-Based Design. International Conference on Computer Safety, Reliability and Security (SafeComp), pp. 58-71, 2008
7. T.P. Kelly, and R. Weaver, "The Goal Structuring Notation – A Safety Argument Notation", in Proc. DSN Workshop on Assurance Cases: Best Practices, Possible Outcomes, and Future Opportunities, 2004.
8. B. Kaiser, P. Liggesmeyer and O. Mackel, A new component concept for fault trees. Proceedings of the 8th Australian workshop on Safety critical systems and software, pp. 37-46, October 01, Canberra, Australia, 2003