SAFEST

Probabilistic Risk Assessment

Dynamic Fault Trees & Reward Event Trees

Static/Dynamic Fault Trees & Reward Event Trees Analysis Tool

www.safest.dgbtek.com

User Manual

Version 2.0.0

Dr. Falak Sher,
Asst. Prof. Matthias Volk,
Prof. Joost-Pieter Katoen,
Prof. Mariëlle Stoelinga

Prepared by
DGB Technologies LLC
Wyckoff, NJ 07481, USA

DGB Technologies is a multimission company specializing in formal verification tools and technologies for stochastic analysis of mission and life-critical systems.

DGB Technologies LLC
Risk & Reliability Analysis
Wyckoff, NJ 07481, USA

Static/Dynamic Fault Trees & Reward Event Trees Analysis Tool
www.safest.dgbtek.com

Dr. Falak Sher,
DGB Technologies LLC
Asst. Prof. Matthias Volk,
Eindhoven University
Prof. Joost-Pieter Katoen,
RWTH Aachen University
Prof. Mariëlle Stoelinga,
Twente University

Abstract

SAFEST is a powerful tool for modeling and analyzing both static and dynamic fault trees as well as reward event trees. Dynamic fault trees (DFTs) extend standard fault trees by providing support for faithfully modeling spare management, functional dependencies, and order-dependent failures. Reward event trees (RETs) extend traditional event trees with state rewards as well as non-deterministic decision-making at states, thus providing upper and lower bounds on the analysis results.

In addition to BDD-based analysis, the SAFEST tool provides an efficient and powerful analysis of DFTs/RETs via probabilistic model checking – a rigorous, automated analysis technique for probabilistic systems. The backbone of the analysis is based on efficient state space generation. Several optimization techniques are incorporated, such as exploiting irrelevant failures, symmetries, and independent modules. Probabilistic model checking allows us to analyze the resulting state space concerning a wide range of measures of interest. In addition, an approximation approach is provided that builds only parts of the state space and allows to refine the computations up to the desired accuracy iteratively.

The SAFEST tool allows embedding DFTs within RETs thus providing probabilities of transition branches of RETs. It provides a graphical user interface for creating, generating, simulating, simplifying, and visualizing the results of fault trees/event trees.

SAFEST is a state-of-the-art analysis tool, as demonstrated by an experimental evaluation and comparison with existing tools. A variety of case studies, including vehicle guidance systems, train operations in railway station areas, and energy systems such as (nuclear) power plants have been done. This document explains how to use the SAFEST.

Contents

Contents
● Introduction
Dynamic Fault Trees (DFTs)
Reward Event Trees (RETs)
Probabilistic Model-checking
● SAFEST Project
Creation
Preferences
Open
Export
Features
Help
Status
● Fault Tree Analysis Module
Fault Trees
Add Fault Tree
Import Fault Trees
Import Fault Trees from SysML v2 Models:
Get the SysML model from a file
Get the SysML model/project via API
Fault Tree Pages
Add Fault Tree Page
Update Fault Tree Page
Labelled Events
CCF Groups
Probabilistic Dependencies
Initial Conditions

- Configurations
  - Parameter Sets
    - View
    - Creation
    - Import
    - Duplicate
    - Export
    - Update
      - Constants
      - Constant Expressions
      - Failure event distributions
      - Empirical failure distributions
  - Attribute Sets
  - Metrics
    - Basic
    - Complex
    - Importance
    - Custom
  - Computing
  - (Exact) Analysis
  - Bounded analysis
  - Graphs
  - Interactive simulation
  - Minimal cut set (for static fault trees)
- Event Tree Analysis
  - Configurations
    - Parameter Sets
      - View
      - Creation
      - Import
      - Duplicate
      - Export
      - Update
        - Constants
        - Constant Expressions
    - Loss Sets
    - Consequence Sets
  - Event Trees
    - View
    - Creation
    - Export

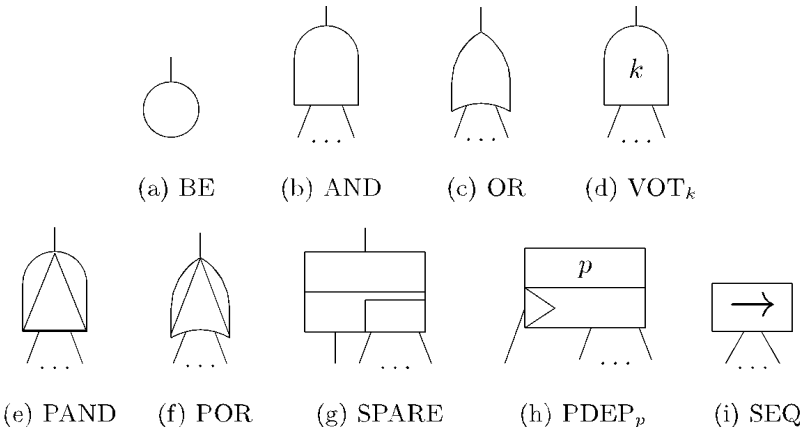
Import
Update
Computing
(Exact) Analysis
Graphs
● Interfaces
Constants
Constant Expressions
FT event
FT event expression
● Annotation of SysML Models with Safety Information
● Grammars
Regular Expressions of Identifiers and Numeric Constants
Context Free Grammar (CFG) of Real Expressions
Context Free Grammar (CFG) of Boolean Expressions
Context Free Grammar (CFG) of Continuous Stochastic Logic (CSL)

Introduction

Probabilistic risk assessment (PRA) is of critical importance to ensure the safe and reliable operation of today's systems in areas such as transportation, infrastructure, power generation, space exploration, etc. Fault trees and event trees are popular models in PRA and are recommended by many standards and regulatory bodies.

Dynamic Fault Trees (DFTs)

While standard (or static) fault trees (SFT) are widely used and well supported by PRA tools, their modeling capabilities are limited. Several extensions to fault trees have been proposed



over the years to overcome these limitations. Dynamic fault trees (DFT) were introduced by Dugan and are one of the most prominent extensions. DFTs introduce new gate types that allow for more faithful modeling by providing explicit support for spare management, functional dependencies, and order-dependent failures. DFT models have been successfully applied for, e.g., aerospace systems, autonomous driving, railway engineering, and analysis of spacecraft via the COMPASS toolset.

Reward Event Trees (RETs)

Reward event trees extend standard event trees with reward models for states as well as non-deterministic decision-making at states. Unlike standard event trees, the analysis of RETs not only provides the probabilities/frequencies of different outcomes/consequences (for an initiating/accidental event) but also provides upper and lower bounds on the results in the case of decision-making. Furthermore, it allows the evaluation of expected values of different quantities of interest e.g. radionuclide emission, etc. Details about event trees can be read at the link.

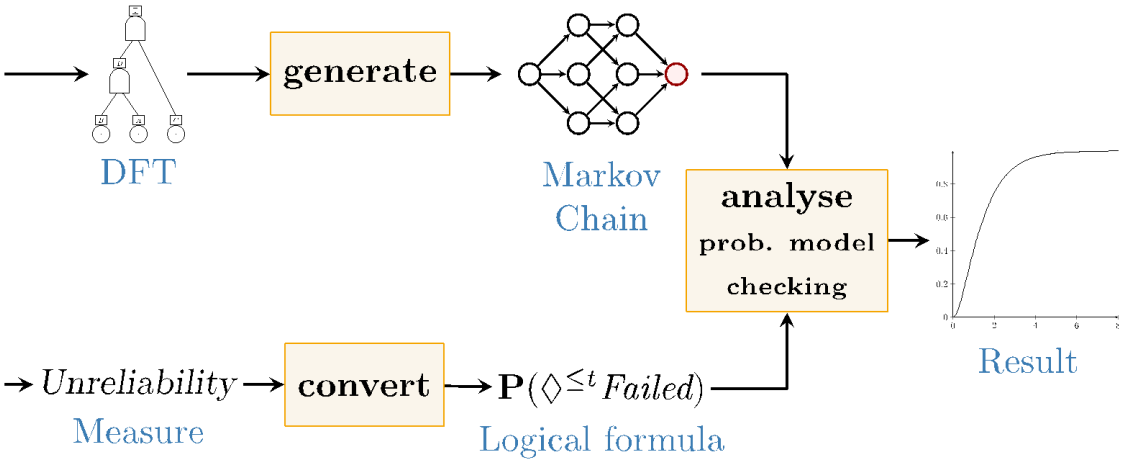
SAFEST is a modern, state-of-the-art tool that allows for modeling and analyzing both standard (static) and dynamic fault trees as well as event trees. SAFEST comes with a web-based graphical interface that allows efficient modeling, visualization, simplification, and interactive simulation of fault trees using a graphical editor. Moreover, it allows embedding DFTs

within RETs thus providing probabilities of transition branches of RETs. The analysis of dynamic fault trees/event trees is enabled via an efficient translation to Markov models and the use of state-of-the-art techniques from probabilistic model checking.

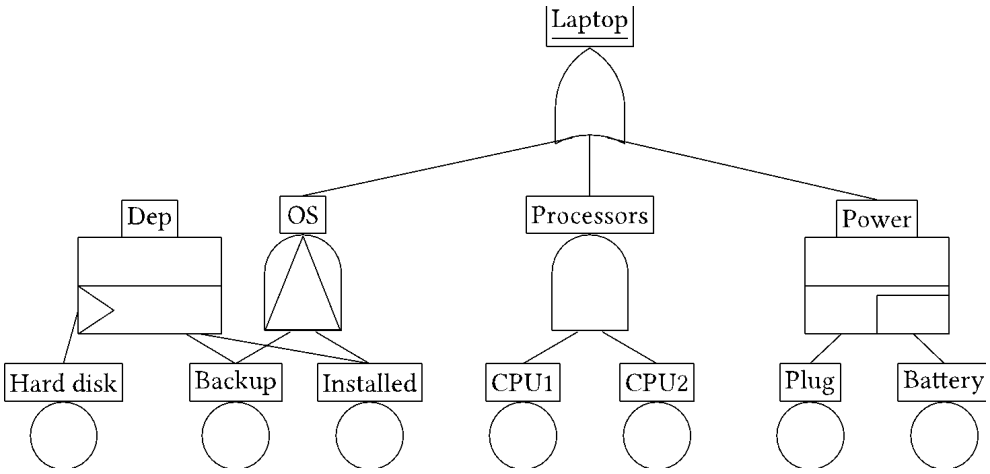
Probabilistic Model-checking

Model checking is a rigorous technique for checking whether a given model satisfies a specification given as a logical formula. Model checking uses highly optimized techniques to efficiently analyze the state space. Probabilistic model checking considers probabilistic systems that capture random behavior, such as Markov models. The approach uses tailored numerical algorithms and answers queries such as “What is the probability of a system failure within a year?” or “What is the expected time to failure when the system has entered a degraded state?”.

We apply probabilistic model checking to DFT/DET analysis. The input is a DFT/DET and a measure of interest, for instance, the unreliability or the mean-time-to-failure (MTTF). From the DFT/DET, a Markov chain is generated. The measure is converted to a logical formula. Both the Markov chain and the logical formula are fed into a probabilistic model checker which computes the results.



The use of probabilistic model checking has several advantages for the DFT/DET analysis. First, model checking supports a wide range of complex logical formulas that can be checked out of the box. Thus, we use a one-for-all analysis approach instead of having to develop algorithms tailored to each type of measure. Second, our approach uses model checking as a black box. This means we can easily change the underlying analysis tools and directly incorporate and benefit from recent advances in model-checking algorithms without changing the overall approach. Third, unlike approaches based on Monte Carlo simulation, probabilistic model checking yields exact results and, in particular, is agnostic to rare events.



The key innovation of our approach is an efficient generation of the state space to combat possible state space explosion. To this end, we developed several optimizations that exploit irrelevant failures of events, symmetric structures, and independent modules in the DFT. Furthermore, we incorporated efficient analysis techniques for static fault trees based on binary decision diagrams (BDD). Finally, we developed an approximation approach based on only building the most relevant parts of the state space. All of these techniques allow for efficient analysis of DFTs.

Our experimental evaluation shows that our tool significantly outperforms existing DFT analysis tools and can handle DFTs with several hundred elements. We have demonstrated the performance of our tool and the modeling capabilities of DFTs in several industrial case studies. Examples include DFT models for vehicle guidance systems and analyzing the impact of infrastructure failures on train operations in railway station areas.

There are two main modules of the SAFEST tool:
Fault Tree Analysis: It allows the building of models of different failure scenarios of systems that may arise during their life cycles.
Event Tree Analysis: This allows the building of event tree models that allow for analysis of the probabilities/frequencies of different outcomes/consequences based on an initiating/accidental event.

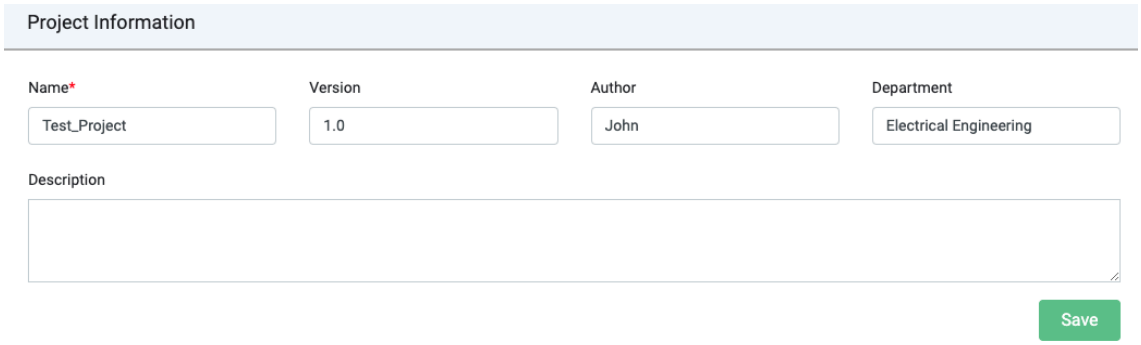
SAFEST Project

In the SAFEST tool, multiple fault tree and event tree models can be created under Fault Tree Analysis and Event Tree Analysis modules.

Creation

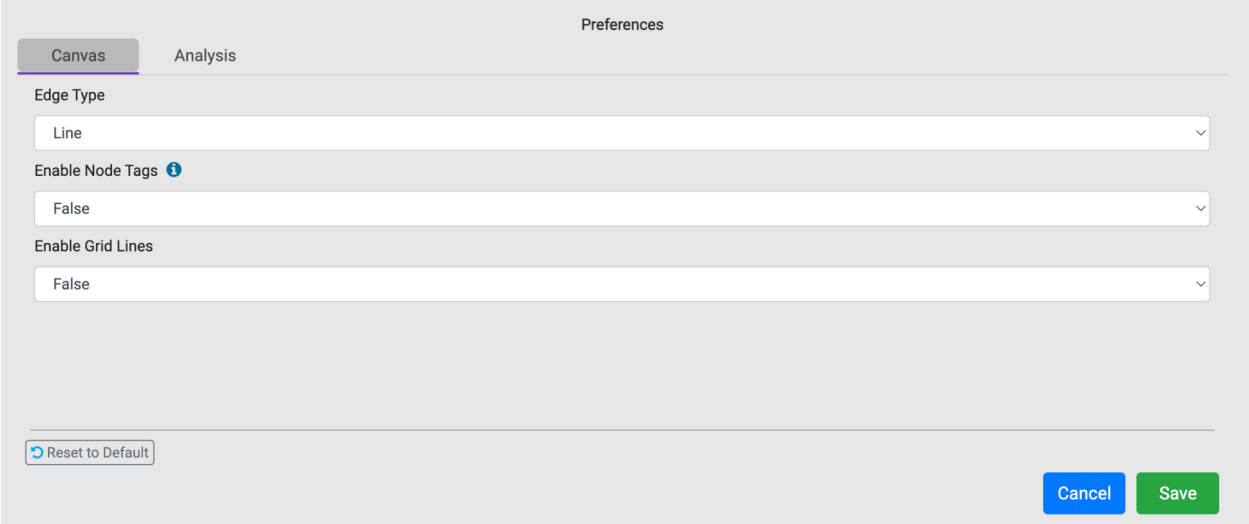
Click on “New Project” in the File menu of the toolbar. The following window will appear.

Fill up mandatory fields, and click the “Save” button. The following page will appear, where users can make any changes if required.

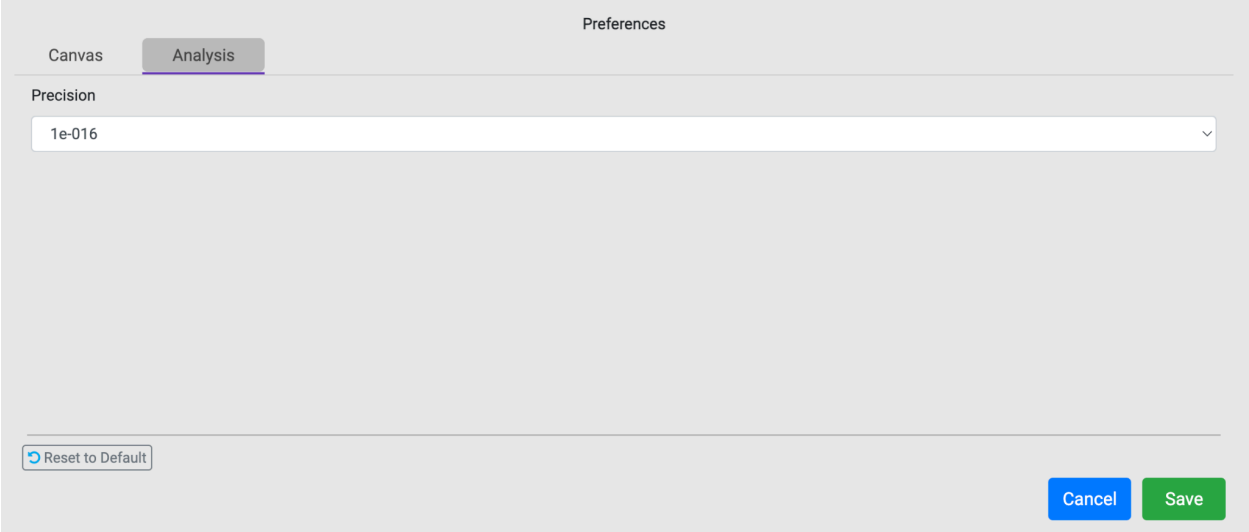


Preferences

Click the “Preferences” in the File menu of the toolbar to set the preferences for your project.



Preferences related to Canvas can be set.



The precision level is set to get the computational results up to the desired level.

Open

Click the “Open Project” in the File menu of the toolbar to open an already existing SAFEST project from your drive.

Export

Click the “Export Project” in the File menu of the toolbar to export the current SAFEST project on your drive.

Features

There are two levels of features that can be enabled for a Project from the Features menu of the toolbar.

- Basic. It is for simple users. Reliability metrics can only be computed for top-level events (TLE or system_failed). Moreover, only basic reliability measures like *reliability/unreliability*, *mean time to failure,* and *average failure probability per hour* can be computed for the TLE.
- Advance. It is for advanced users or researchers. This view gives the full functionality of the SAFEST tool.

Help

In the Help menu of the toolbar, we have:

- Documentation: It contains a link to the user manual of the SAFEST tool, as well as the grammar for expressions used in the models as parameters.
- Activation key: Here you can add a license key and activate the SAFEST tool functionality.

Status

In the status bar, the information about the view of the currently selected project is shown along with the status of the Analysis engine, which is either Running or Stopped.

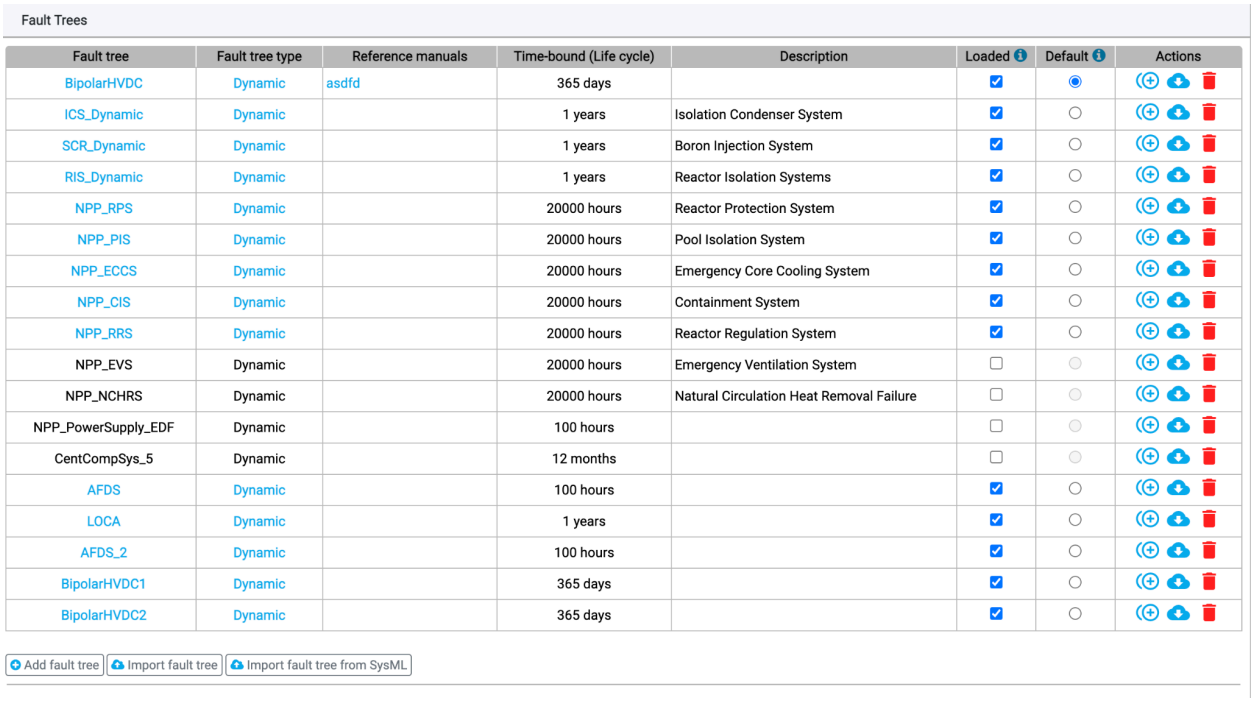
Fault Tree Analysis Module

It contains three sub-sections:
Fault Trees: It contains all fault trees in the project.
Configurations: It contains parameter sets and attribute sets that are used in the fault tree. It also contains metrics that are analyzed on fault trees in the computing section/
Computing: It contains all methods that are used to analyze fault trees: Analysis, Bounded Analysis, Graphs, Interactive Simulation, and Minimal Cut Sets.

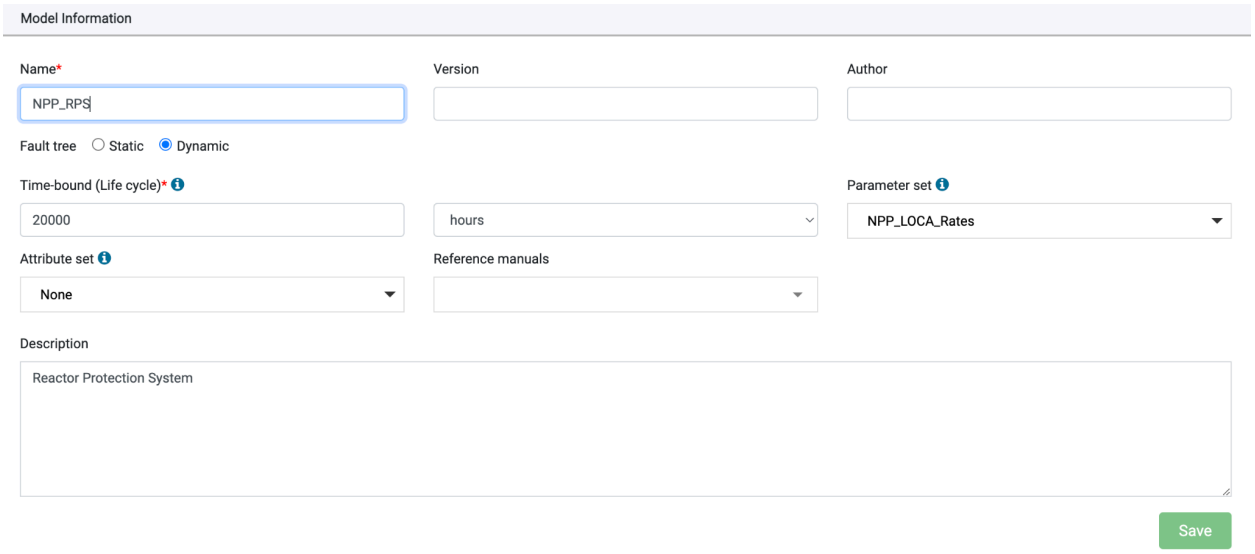
Fault Trees

This section contains a list of all fault trees in a project. Each fault tree has sub-sections: Fault Tree Pages, Labelled Events, CCF Groups, Probabilistic Dependencies, and Initial Conditions.

Click on the “Fault Trees” under “Fault Tree Analysis” in the left panel to display all fault trees in a table.
Default: A fault tree that is worked upon the most can be selected as a default model by selecting the corresponding radio button.
Load: One can load/unload fault trees from the list to improve the performance. Note only fault trees that are loaded can be worked upon.
Copy: Click on the plus button in the Actions table to create a copy of the corresponding fault tree.
Download: Click on the download icon to download the corresponding fault tree.



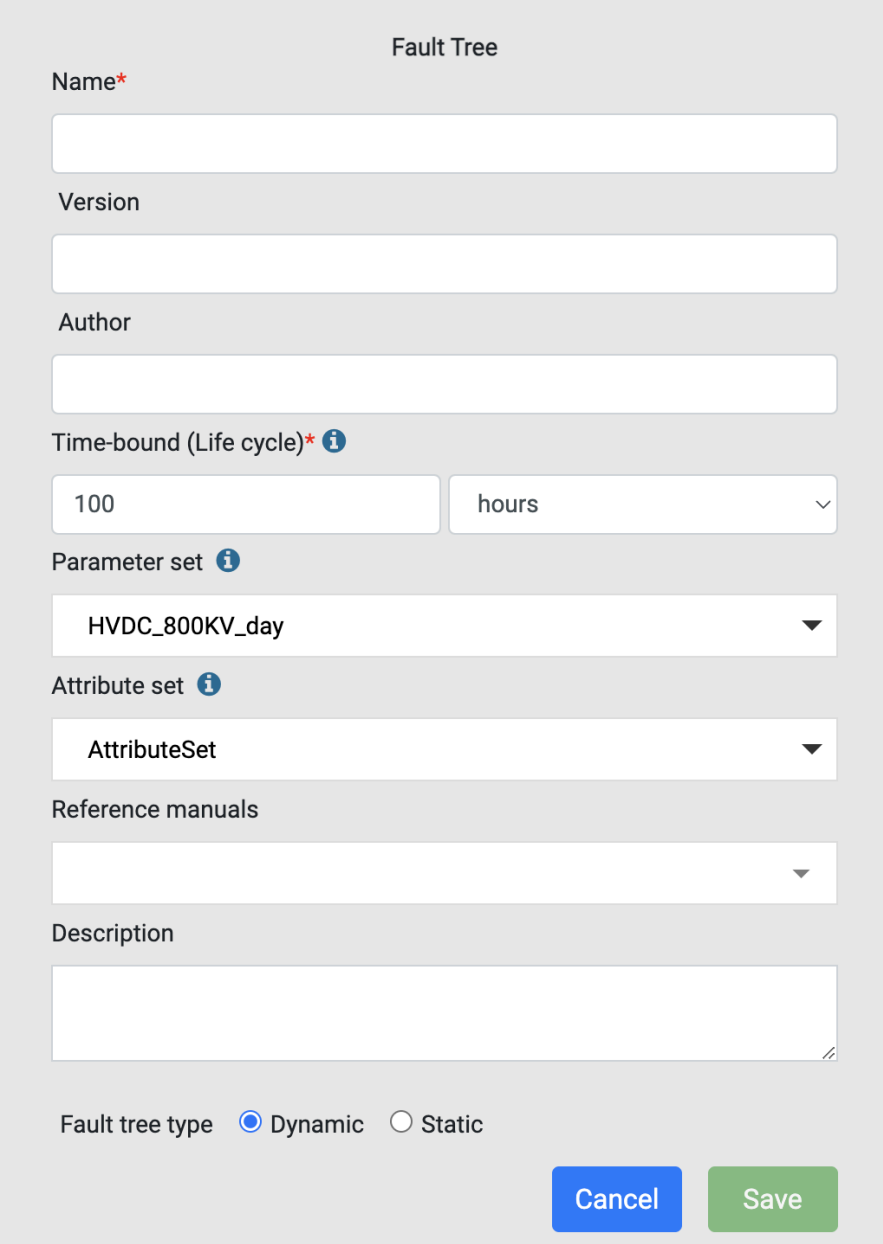
Click on the fault tree name in the left panel/table to see details about the model.



Change fields and click the “Save” button.

Add Fault Tree

Click on the “Add fault tree” button on the “Fault Trees” page to create a new fault tree.

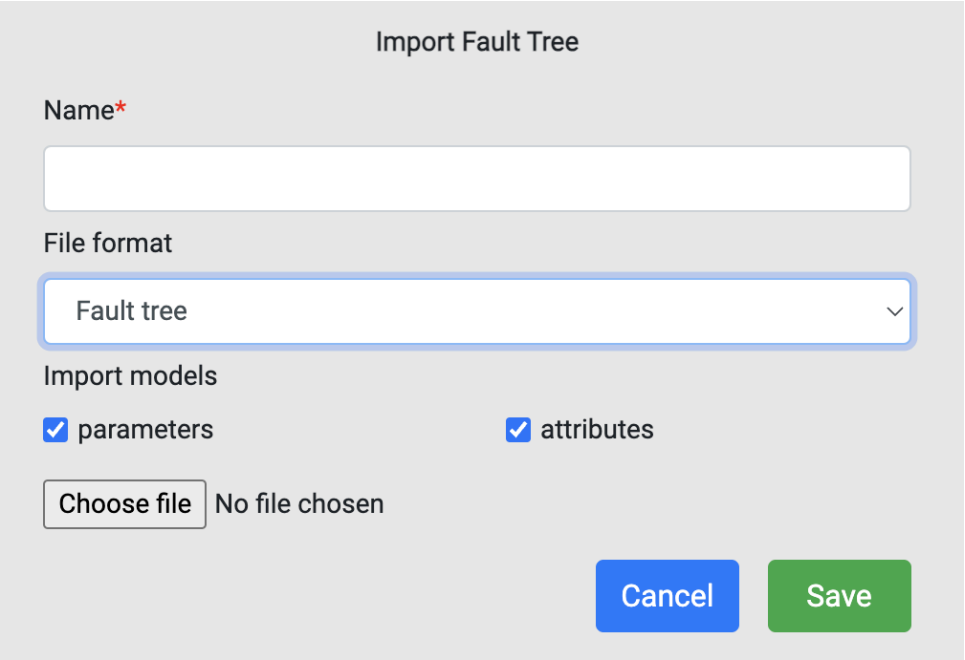


- A time bound for which the model is to be analyzed may be inserted. This value can be changed at the time of analysis as well.
- A parameter set in which parameters to be used in the model are defined may be selected. It can also be changed at the time of analysis.
- An attribute set can be attached to a fault tree. It allows the annotation of additional information about the elements of fault trees. For example, all motors in a system can be attached “Motor” attribute.
- A list of reference manuls can also be attached with a fault tree.
- The type of fault tree can be selected as “Static” or “Dynamic”.

Fill up the mandatory fields and click the “Save” button.

Import Fault Trees

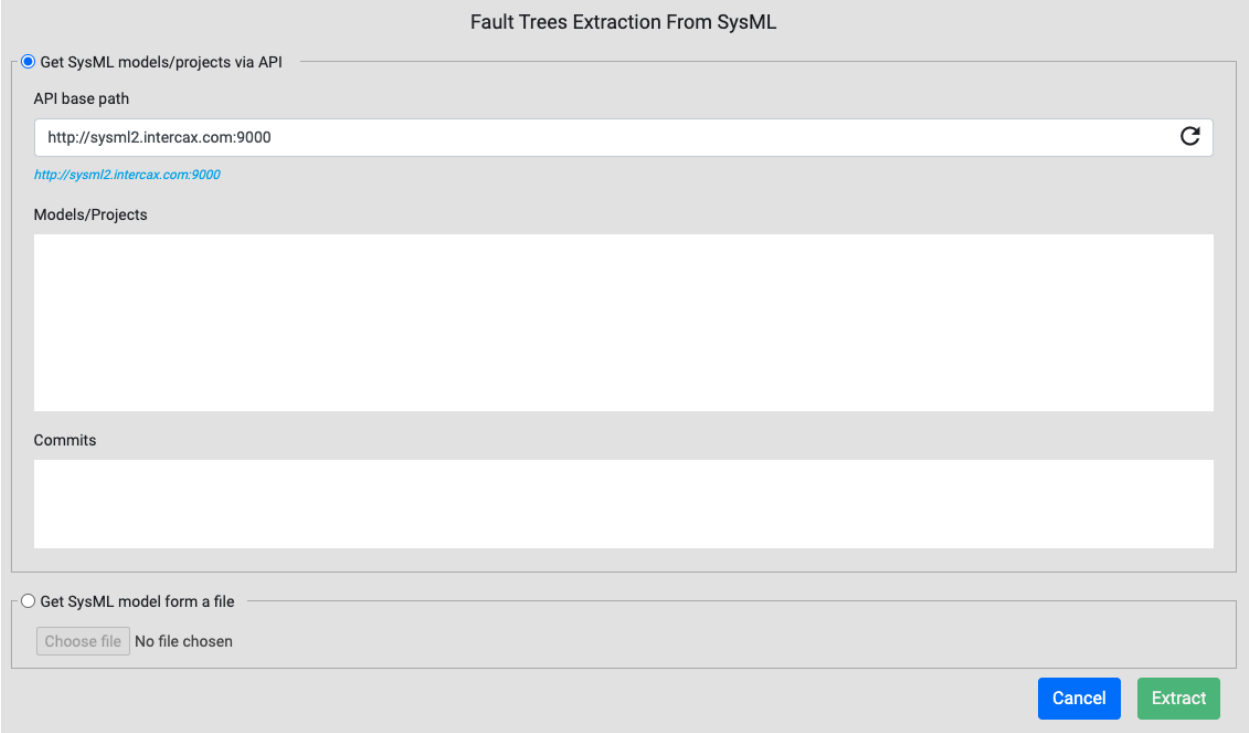
Click on the “Import failure model” button on the “Fault Trees” page to import an already created fault tree.



Select the format of the model and a file from the drive. Click the “Save” button. We support “JSON” “Galileo” and “Fault Tree” formats. Note that Fault Tree (.fm) is the format of the SAFEST tool. If the selected model has parameters/attributes, one can decide whether they are to be imported or not.

Import Fault Trees from SysML v2 Models:

Users have the option to extract failure models from SysML 2.0 models. The SysML model has to be annotated with safety information in order to generate DFTs automatically out of it. Click the “Import failure model from SysML” button on the “Failure Models” page.



The user has two options to extract DFTs out of SysML models

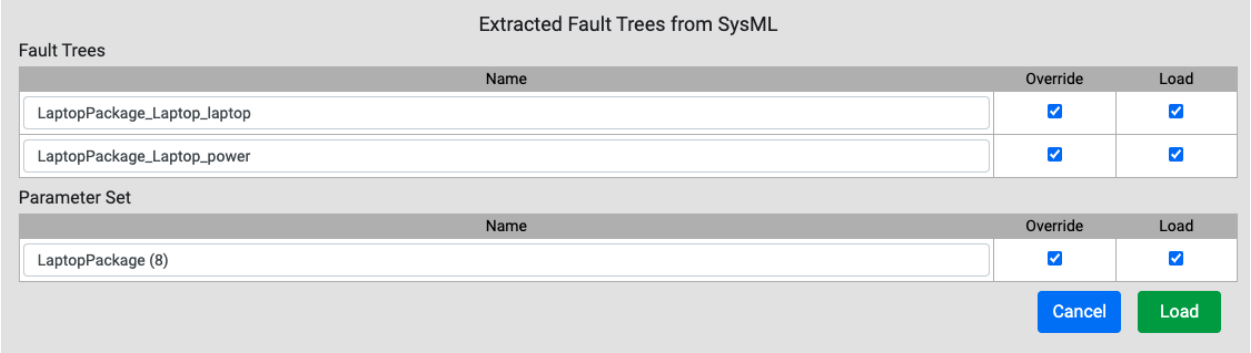
Get the SysML model from a file

Compile a SysML model, which is annotated with safety information – please read the **Annotate SysML Models with Safety Information section** below for further details on how to annotate SysML models with safety information to prepare them for automatic extraction of DFTs out of them, in a Jupyter notebook. And run the following command to export the model in JSON format. **Currently, we support the latest version – v0.45.0 – of SysML 2.0:** SysML-v2-Release

```
%export <package_name>
```

After downloading, DFTs can be extracted from it. Select the radio button “Get SysML model from a file”, select the downloaded file (in JSON format), and click the “Extract” button.

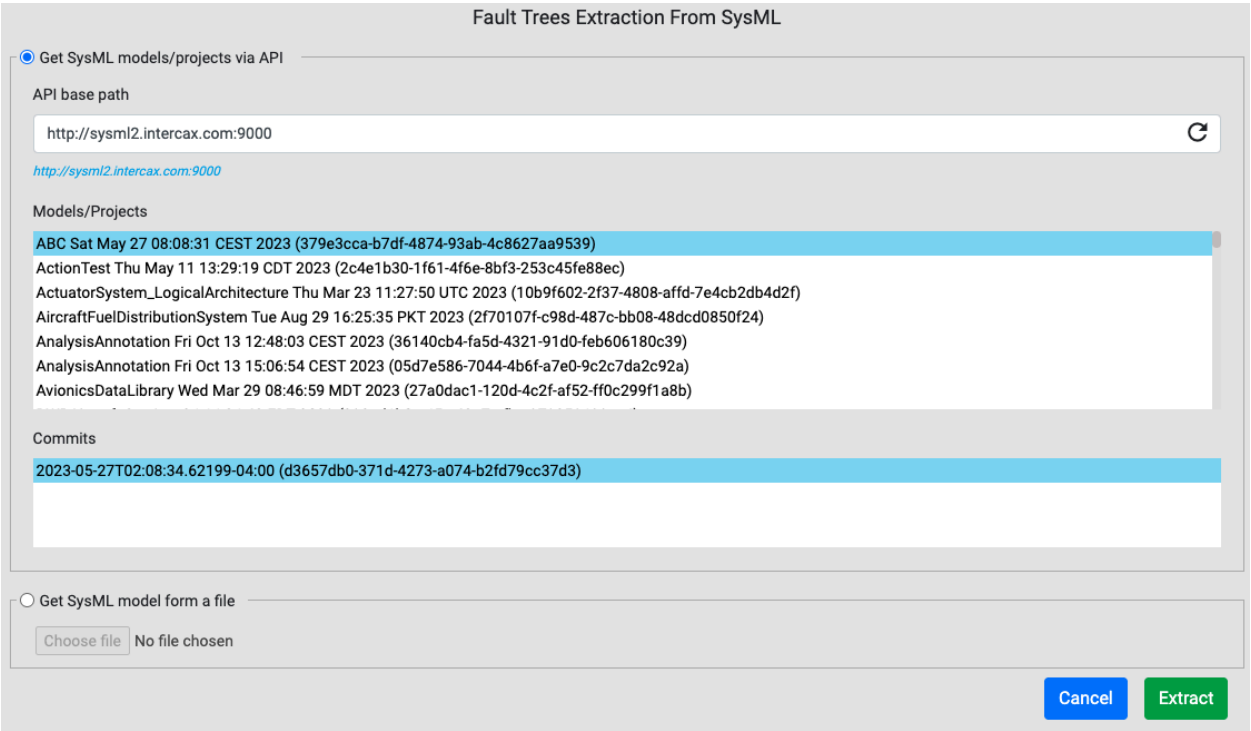
The following popup will appear that contains fault trees of all failure scenarios that have been mentioned in the SysML model. It also contains parameters that have been given in the SysML mode e.g. failure rates of components inside the SysML.



The user can select the failure models as well as parameters to be included in the project.

Get the SysML model/project via API

SysML projects are normally uploaded to some central repository. One has the option to connect to that repository using its API and upload the model from there. Select the “Get SysML models/projects via API” radio button, enter a path in the “API base path” and click the circular arrow.

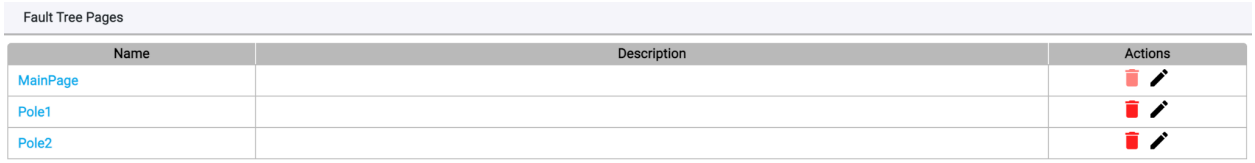


Select a project and its commit, and then click the “Extract” button to upload the SysML model. The rest of the steps are the same as above. For testing purposes, we have uploaded our “LaptopPackage” project to the above repository.

The algorithm will automatically generate fault trees and show them on the “Failure Models” page. Please read **Annotate SysML Models with Safety Information section** below for further details.

Fault Tree Pages

A fault tree may consist of multiple pages. Click on the “Fault Tree Pages” to view a list of pages for the corresponding fault tree.

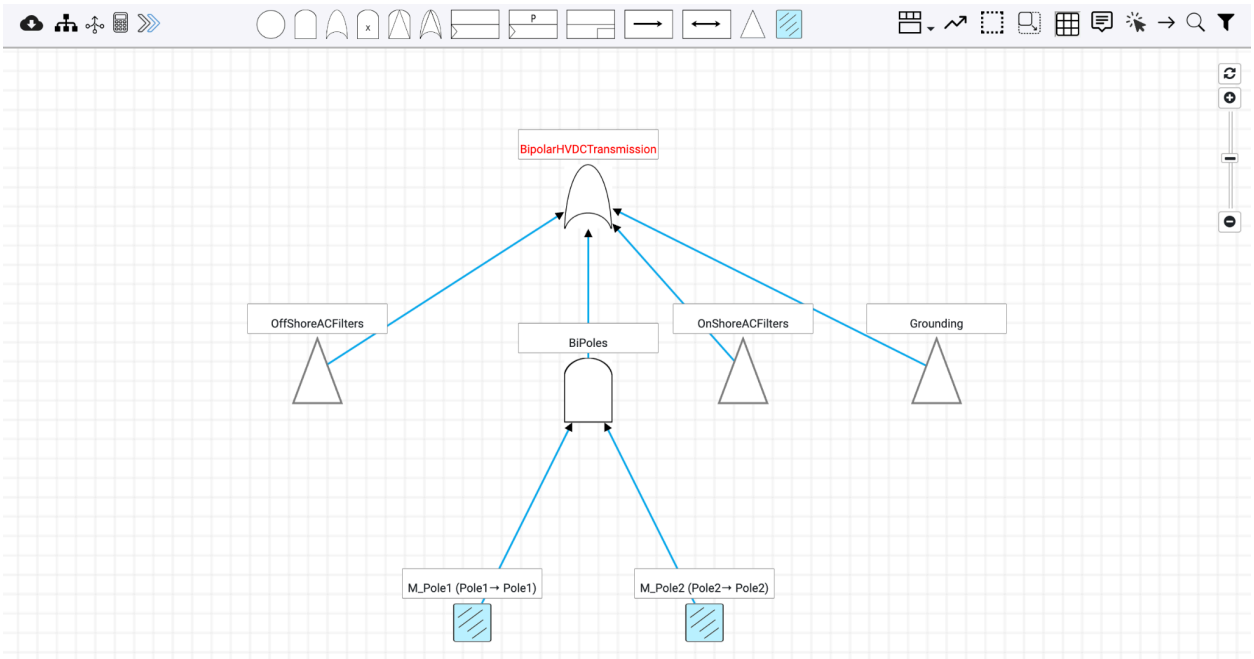


Add Fault Tree Page

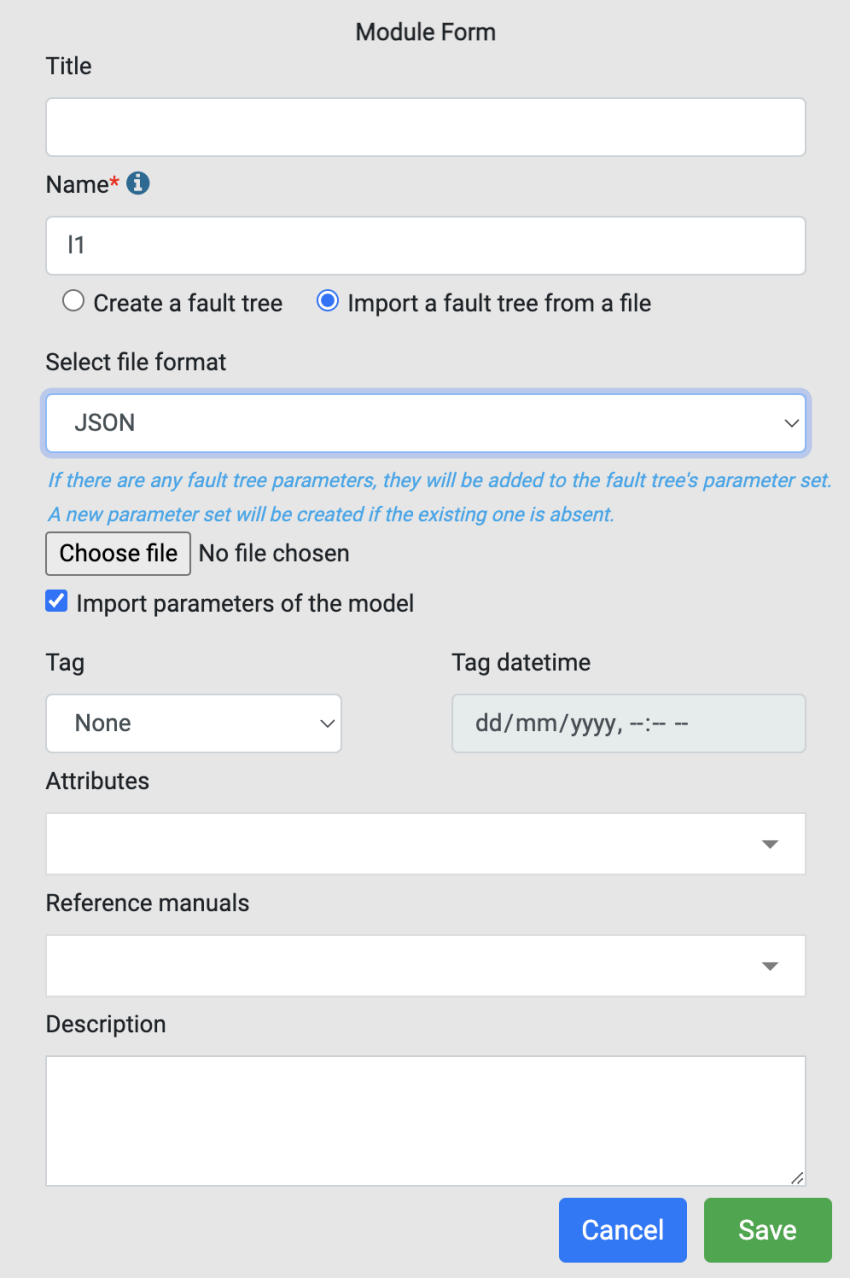
Click on the “Add page” to generate a new fault tree page.

Update Fault Tree Page

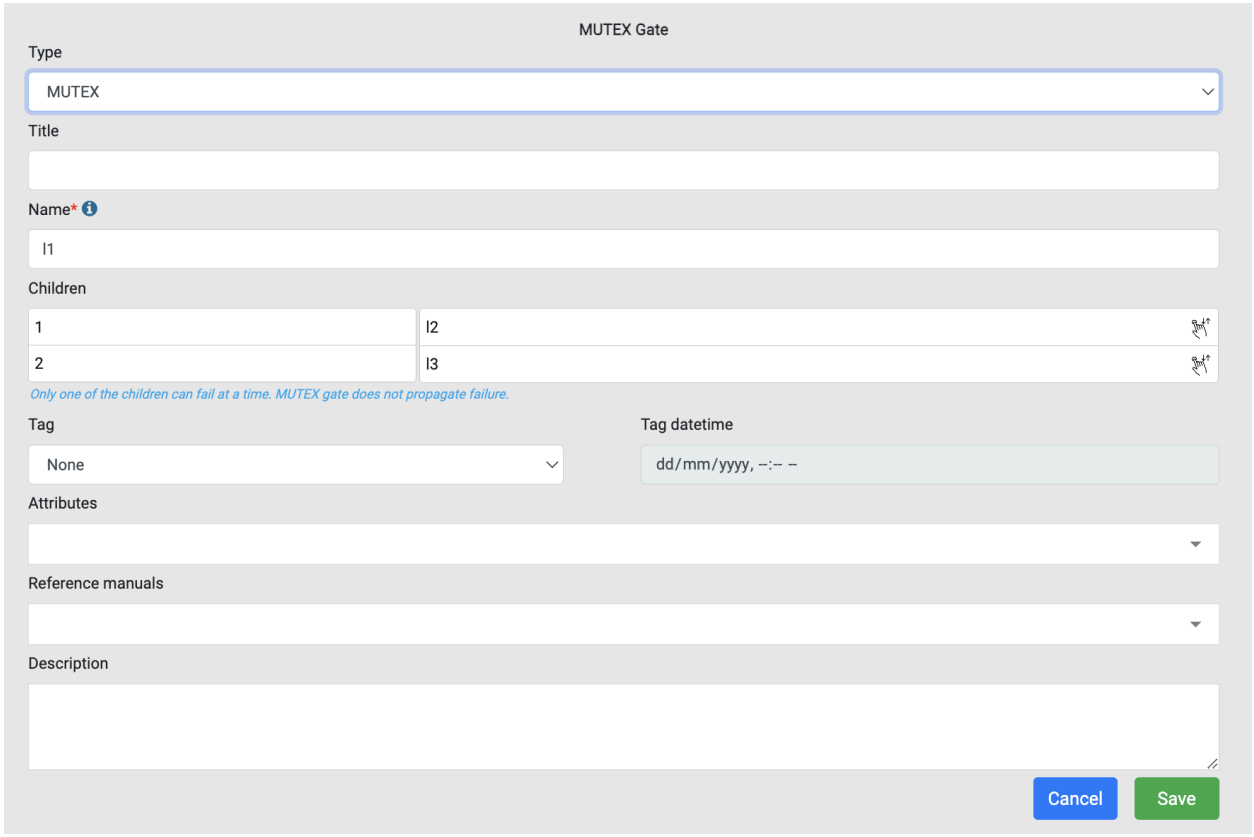
Click on a fault tree page to open a drag-and-drop canvas.

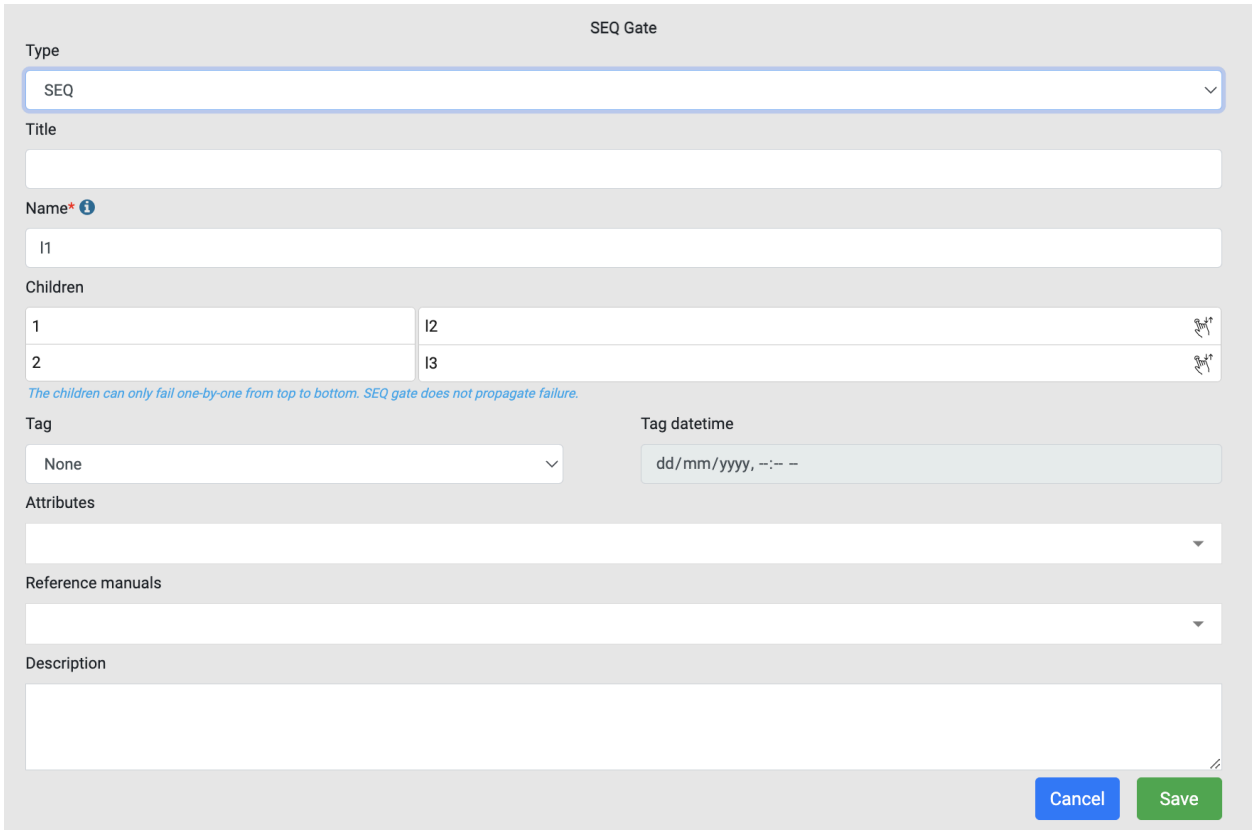


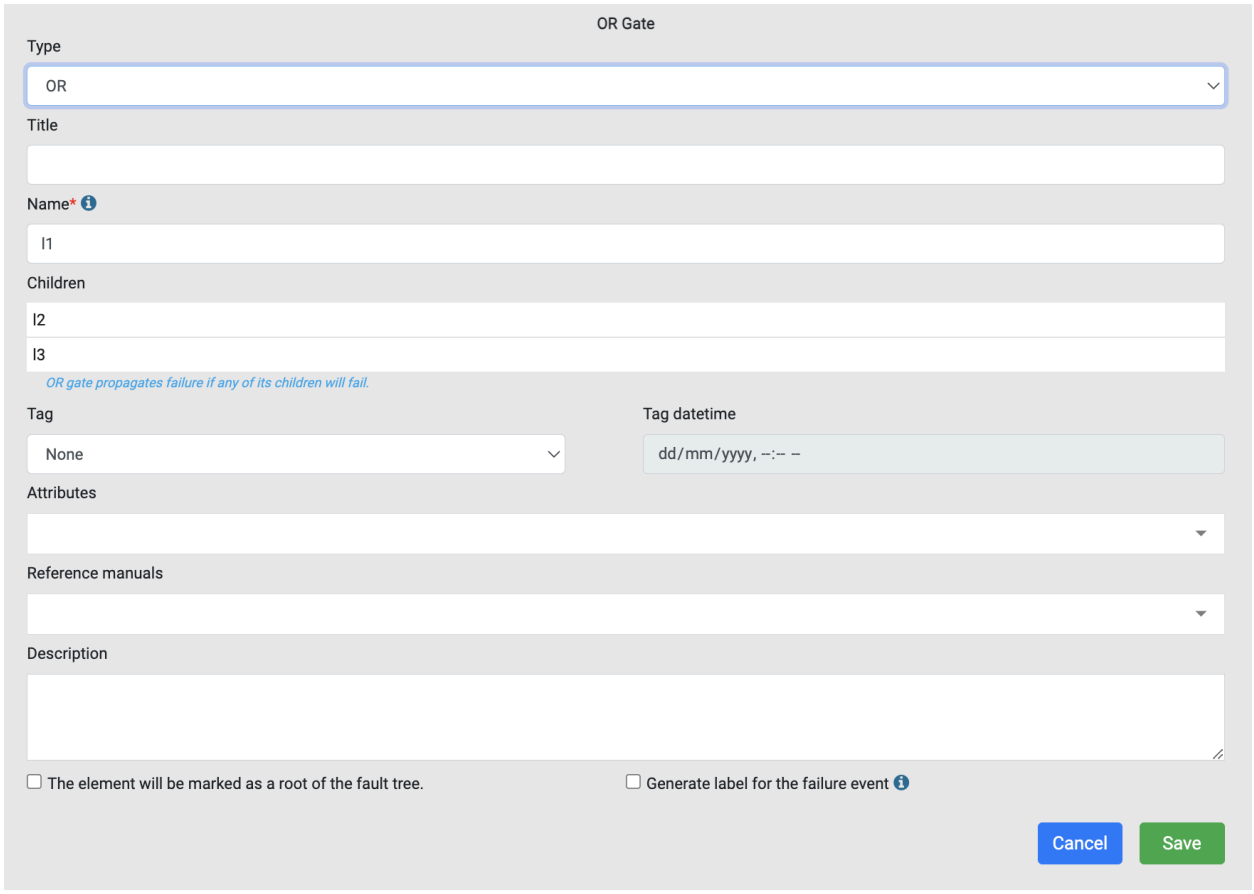
- Draw your tree by dragging different elements from the toolbar.
- Click on an element to update its information in the corresponding popup window

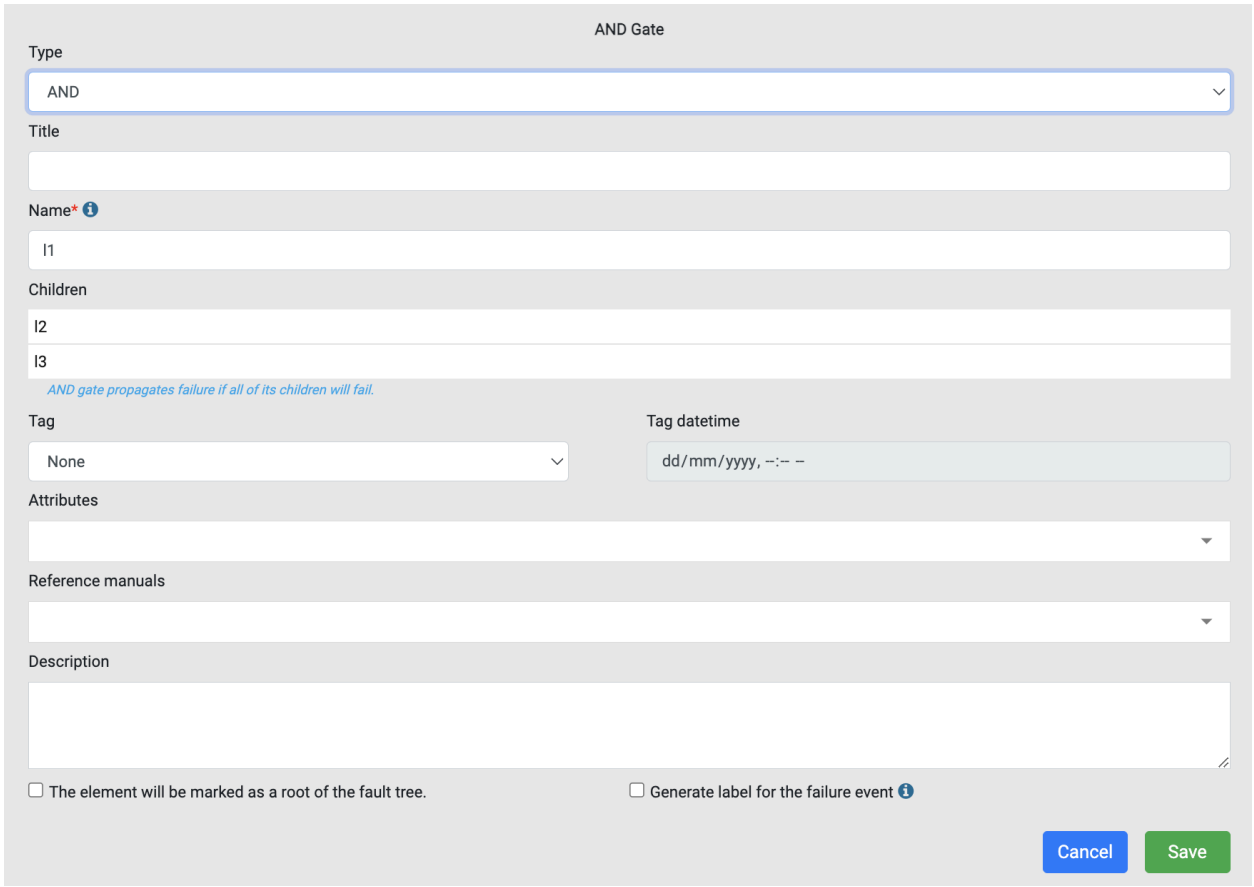


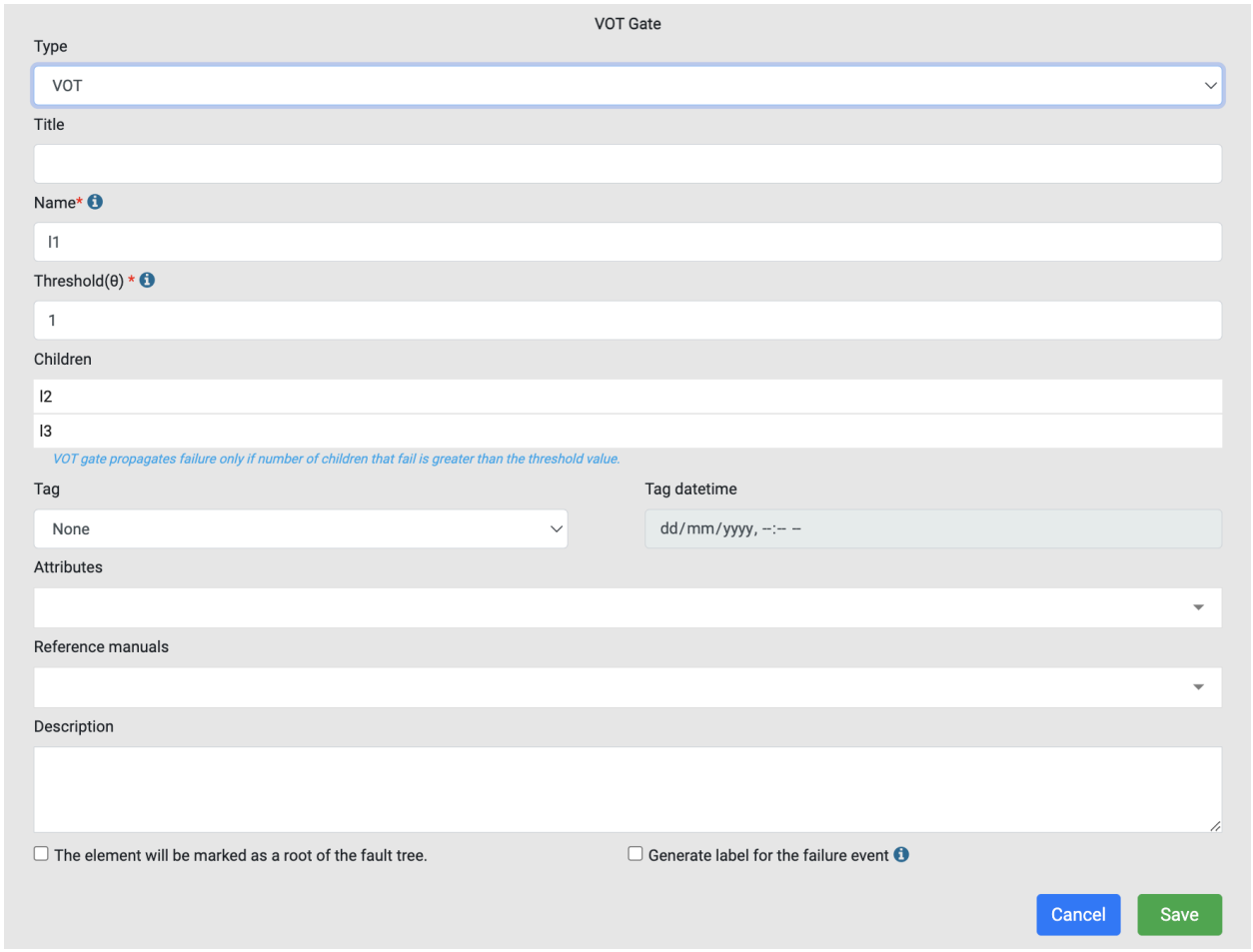
- A module can be created from scratch or uploaded from the drive. To upload, select the file (in JSON or Galileo format) and click the save button.

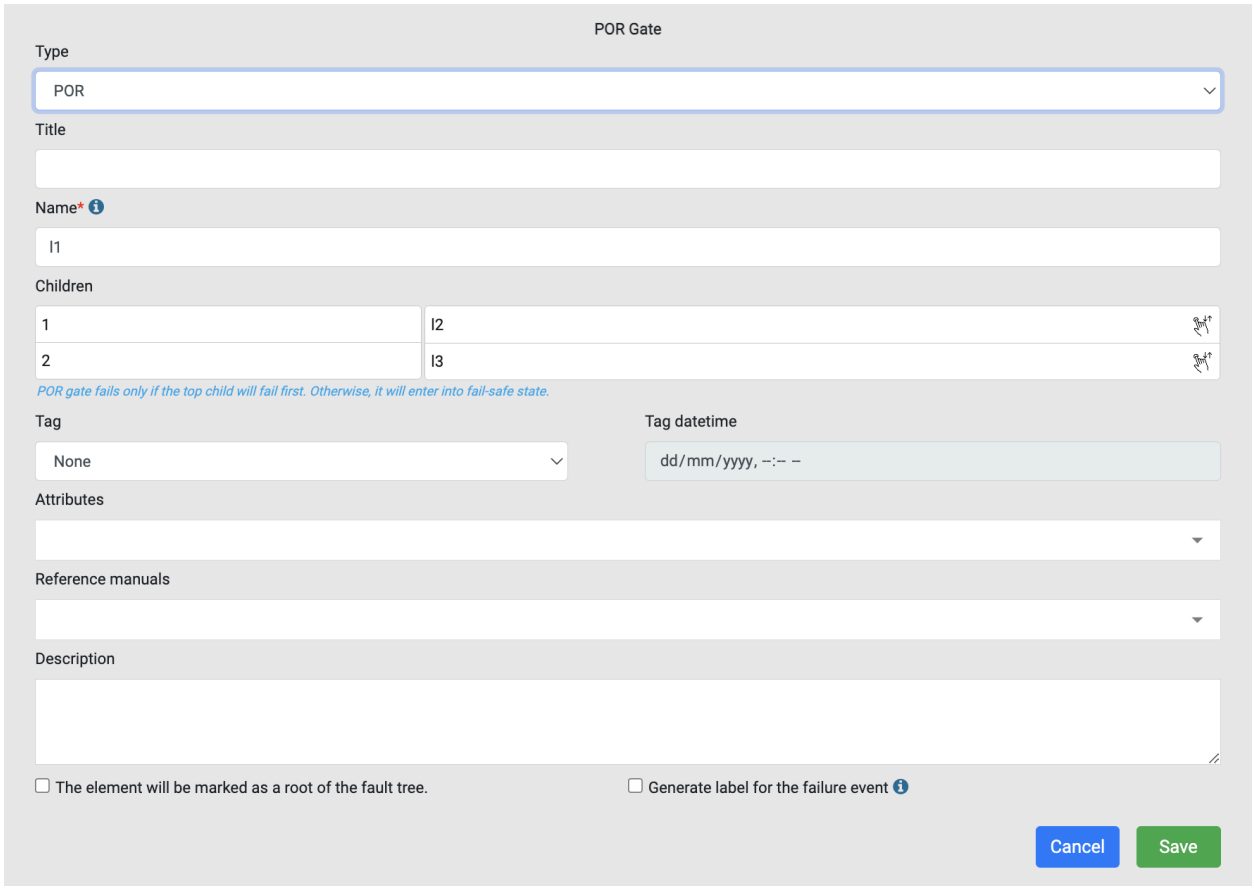

MUTEX Gate
Type
MUTEX
Title
Name*
I1
Children
1
I2
2
I3
Only one of the children can fail at a time. MUTEX gate does not propagate failure.
Tag
None
Tag datetime
dd/mm/yyyy, --:--
Attributes
Reference manuals
Description
Cancel
Save

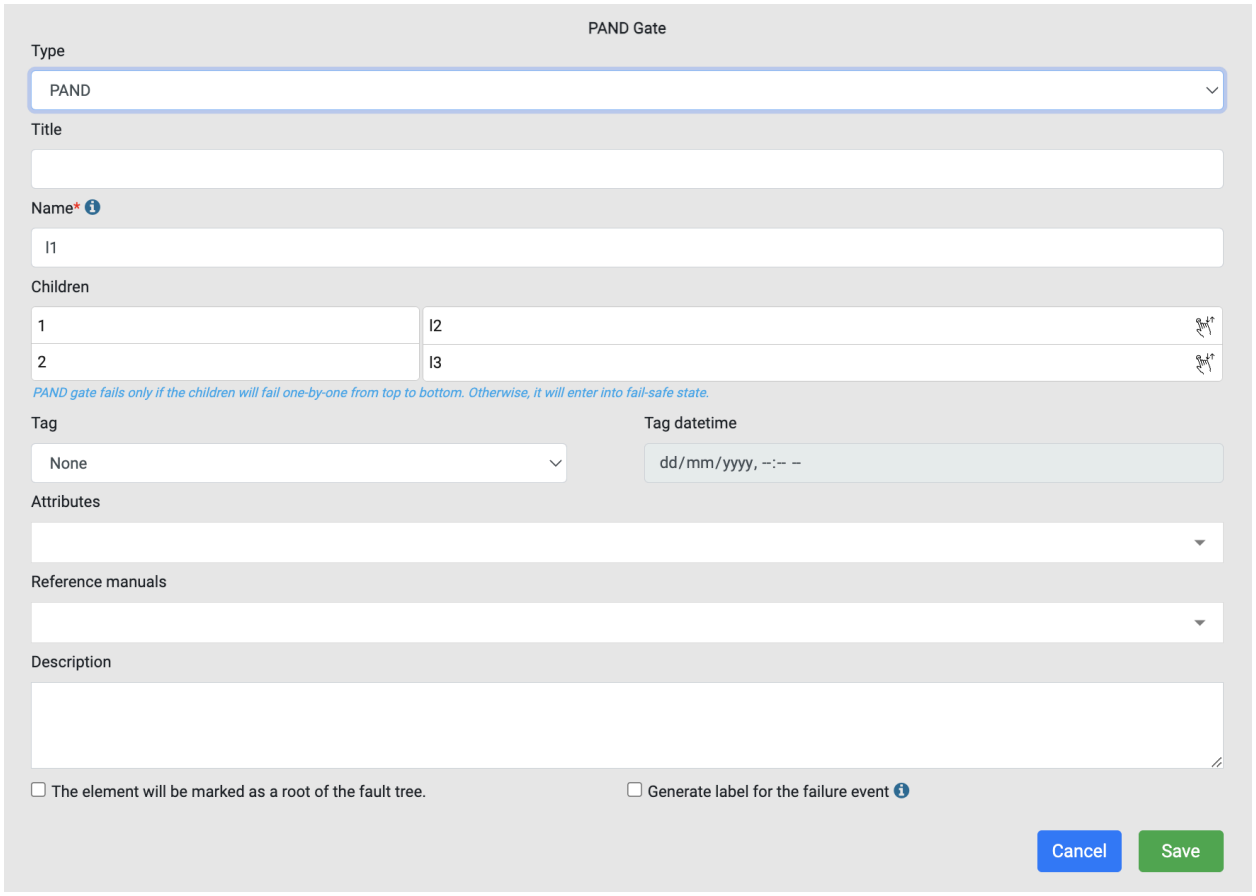

SEQ Gate
Type
SEQ
Title
Name*
I1
Children
1
I2
2
I3
The children can only fail one-by-one from top to bottom. SEQ gate does not propagate failure.
Tag
None
Tag datetime
dd/mm/yyyy, --:--
Attributes
Reference manuals
Description
Cancel
Save

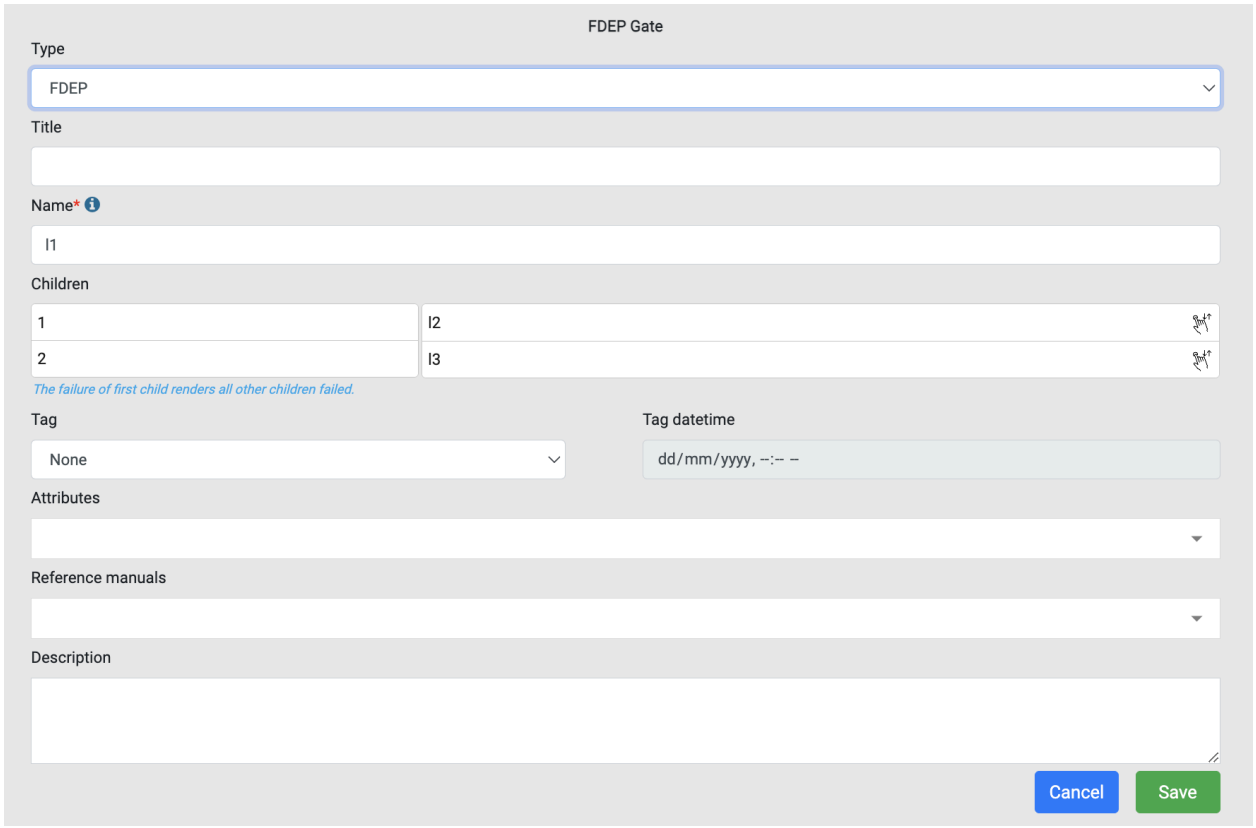

OR Gate
Type
OR
Title
Name*
I1
Children
I2
I3
OR gate propagates failure if any of its children will fail.
Tag
None
Tag datetime
dd/mm/yyyy, --:-- --
Attributes
Reference manuals
Description
The element will be marked as a root of the fault tree.
Generate label for the failure event
Cancel
Save

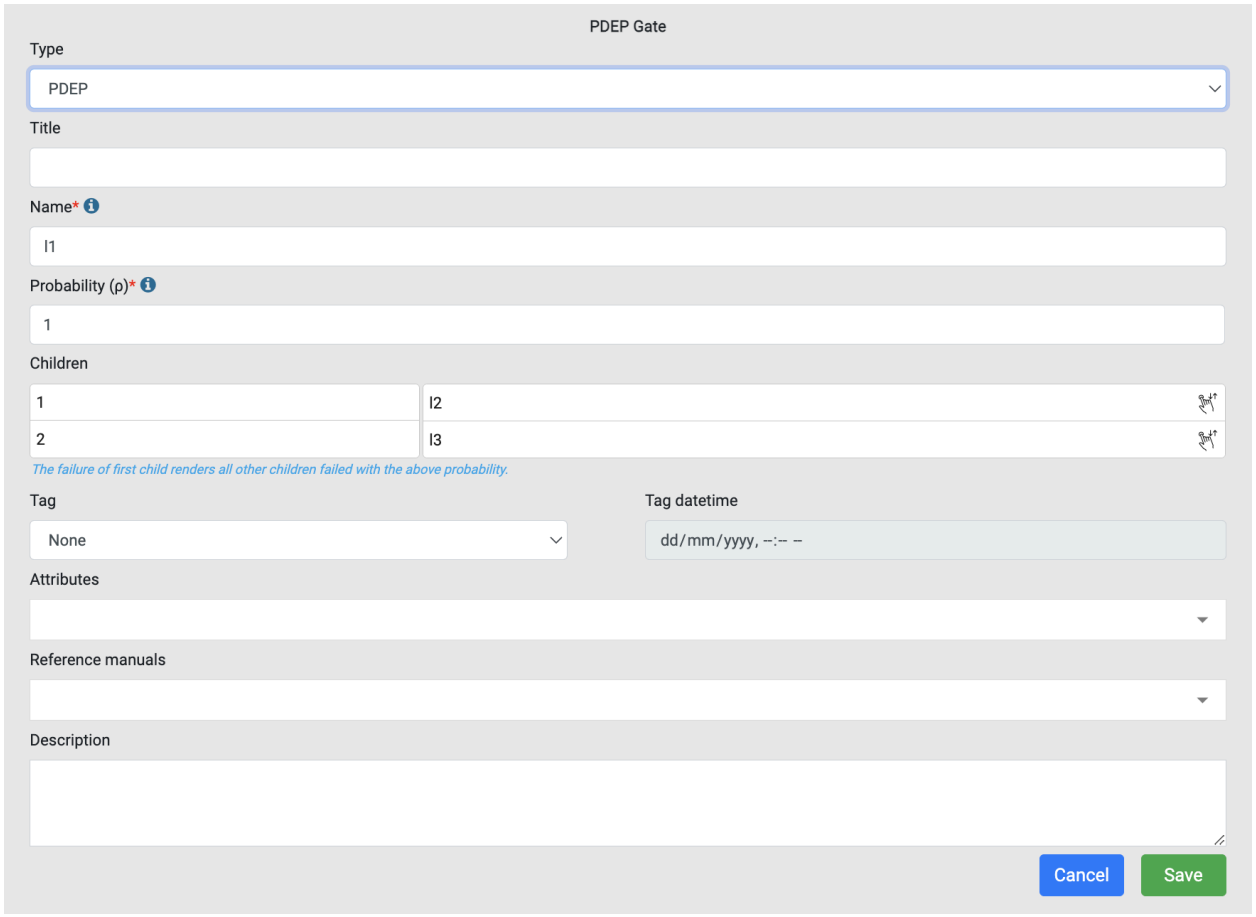

AND Gate
Type
AND
Title
Name*
I1
Children
I2
I3
AND gate propagates failure if all of its children will fail.
Tag
None
Tag datetime
dd/mm/yyyy, --:-- --
Attributes
Reference manuals
Description
The element will be marked as a root of the fault tree.
Generate label for the failure event
Cancel
Save

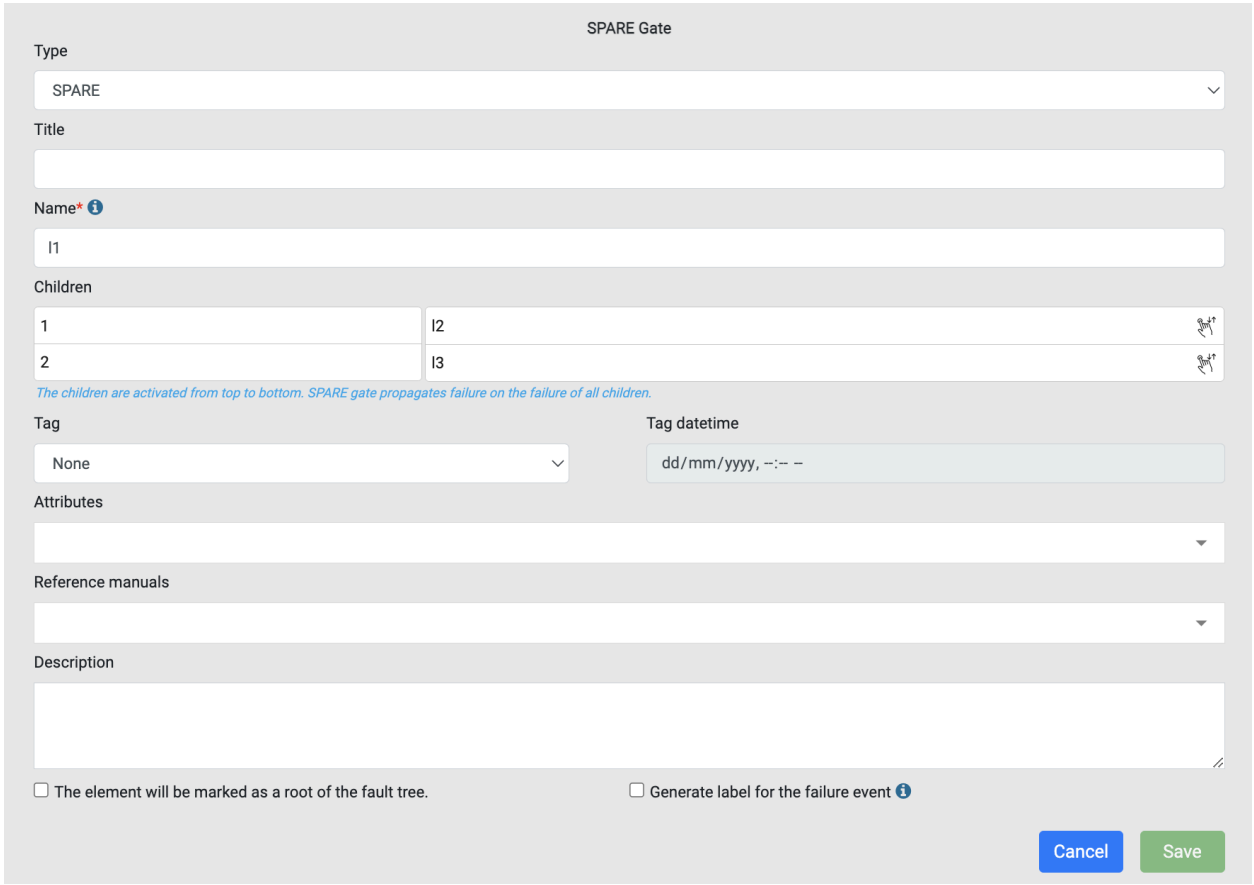

VOT Gate
Type
VOT
Title
Name*
I1
Threshold(θ) *
1
Children
I2
I3
VOT gate propagates failure only if number of children that fail is greater than the threshold value.
Tag
None
Tag datetime
dd/mm/yyyy, --:-- --
Attributes
Reference manuals
Description
The element will be marked as a root of the fault tree.
Generate label for the failure event
Cancel
Save

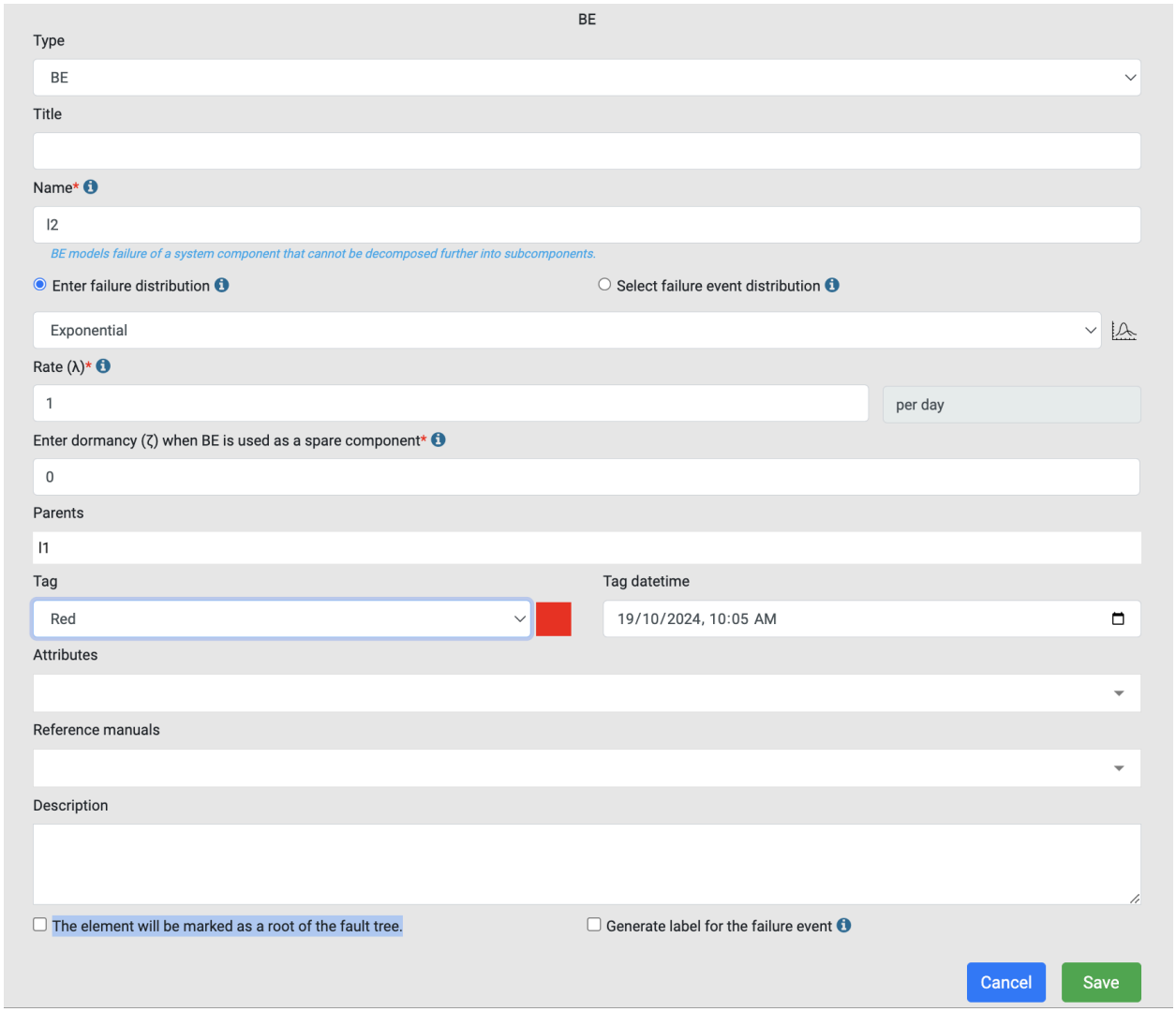

POR Gate
Type
POR
Title
Name*
I1
Children
1
I2
2
I3
POR gate fails only if the top child will fail first. Otherwise, it will enter into fail-safe state.
Tag
None
Tag datetime
dd/mm/yyyy, --:-- --
Attributes
Reference manuals
Description
The element will be marked as a root of the fault tree.
Generate label for the failure event
Cancel
Save

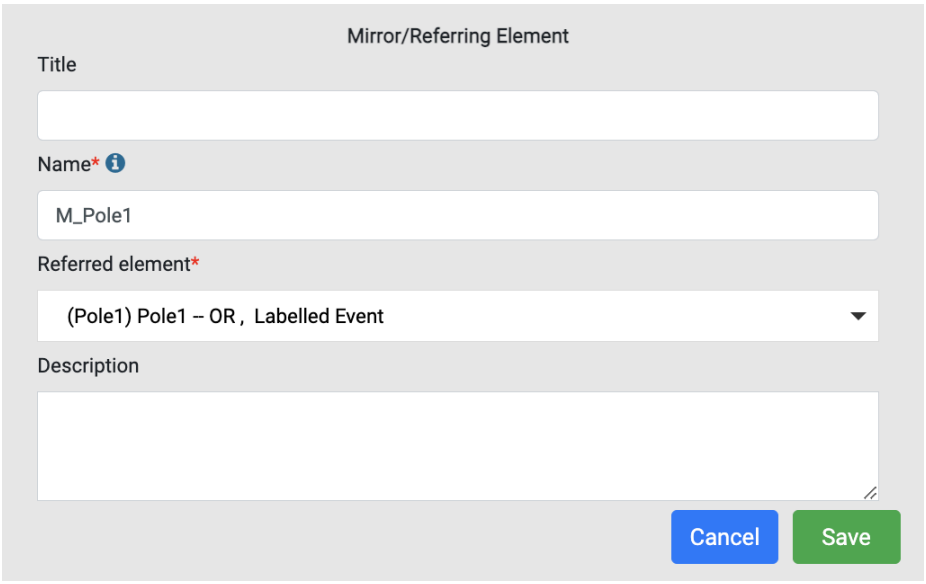

PAND Gate
Type
PAND
Title
Name*
I1
Children
1
I2
2
I3
PAND gate fails only if the children will fail one-by-one from top to bottom. Otherwise, it will enter into fail-safe state.
Tag
None
Tag datetime
dd/mm/yyyy, --:-- --
Attributes
Reference manuals
Description
The element will be marked as a root of the fault tree.
Generate label for the failure event
Cancel
Save


FDEP Gate
Type
FDEP
Title
Name*
I1
Children
1
I2
2
I3
The failure of first child renders all other children failed.
Tag
None
Tag datetime
dd/mm/yyyy, --:-- --
Attributes
Reference manuals
Description
Cancel
Save


PDEP Gate
Type
PDEP
Title
Name*
I1
Probability (p)*
1
Children
1
I2
2
I3
The failure of first child renders all other children failed with the above probability.
Tag
None
Tag datetime
dd/mm/yyyy, --:-- --
Attributes
Reference manuals
Description
Cancel
Save

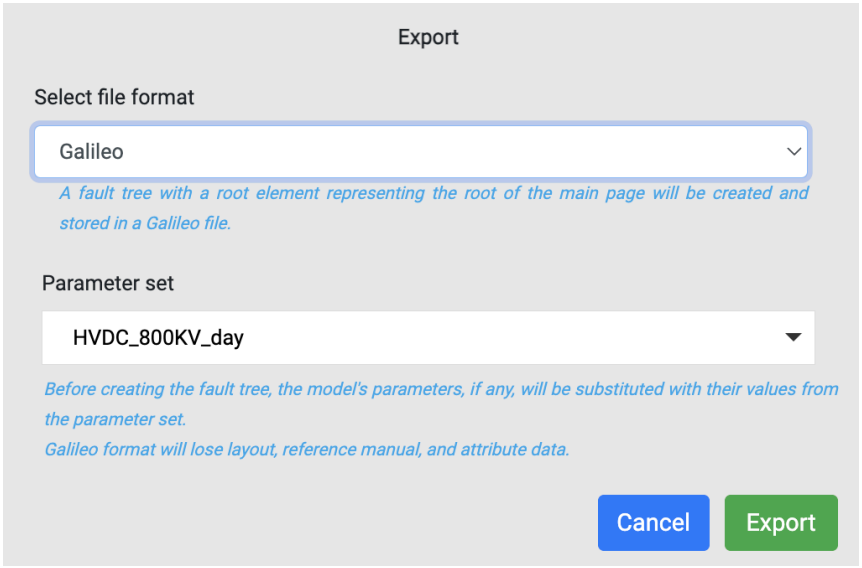

SPARE Gate
Type
SPARE
Title
Name*
I1
Children
1
I2
2
I3
The children are activated from top to bottom. SPARE gate propagates failure on the failure of all children.
Tag
None
Tag datetime
dd/mm/yyyy, --:-- --
Attributes
Reference manuals
Description
The element will be marked as a root of the fault tree.
Generate label for the failure event
Cancel
Save

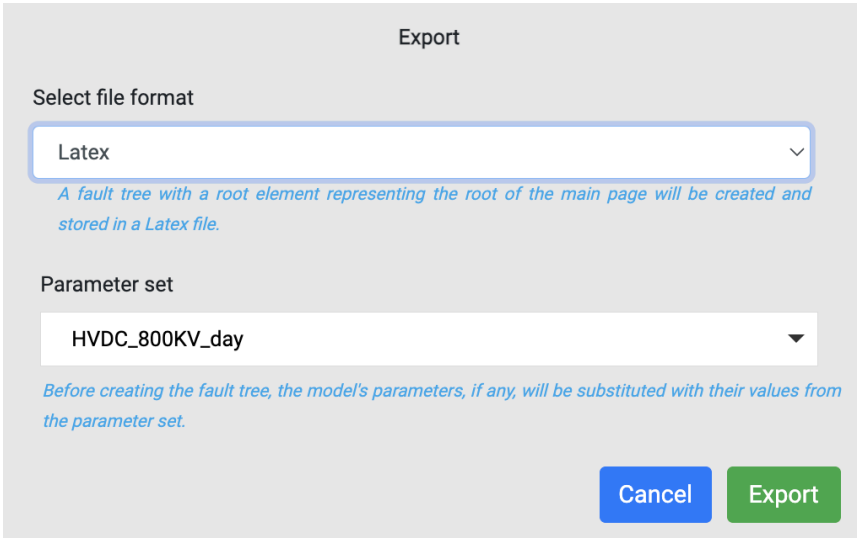


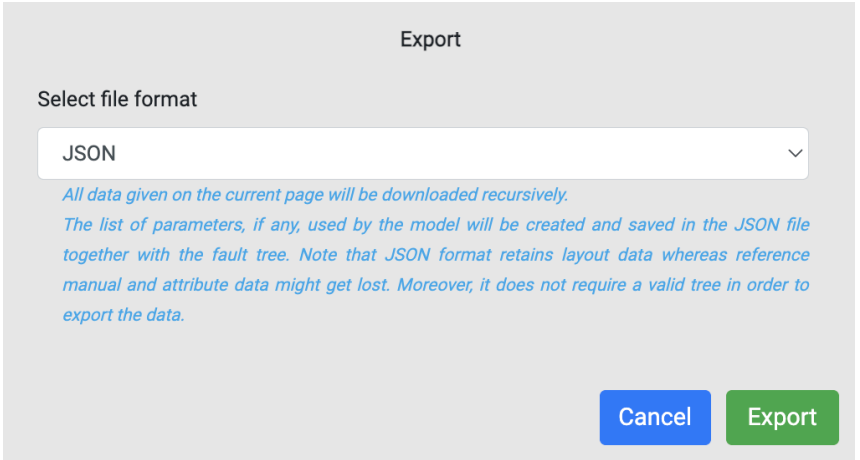


- Elements of a fault tree can be on different pages.

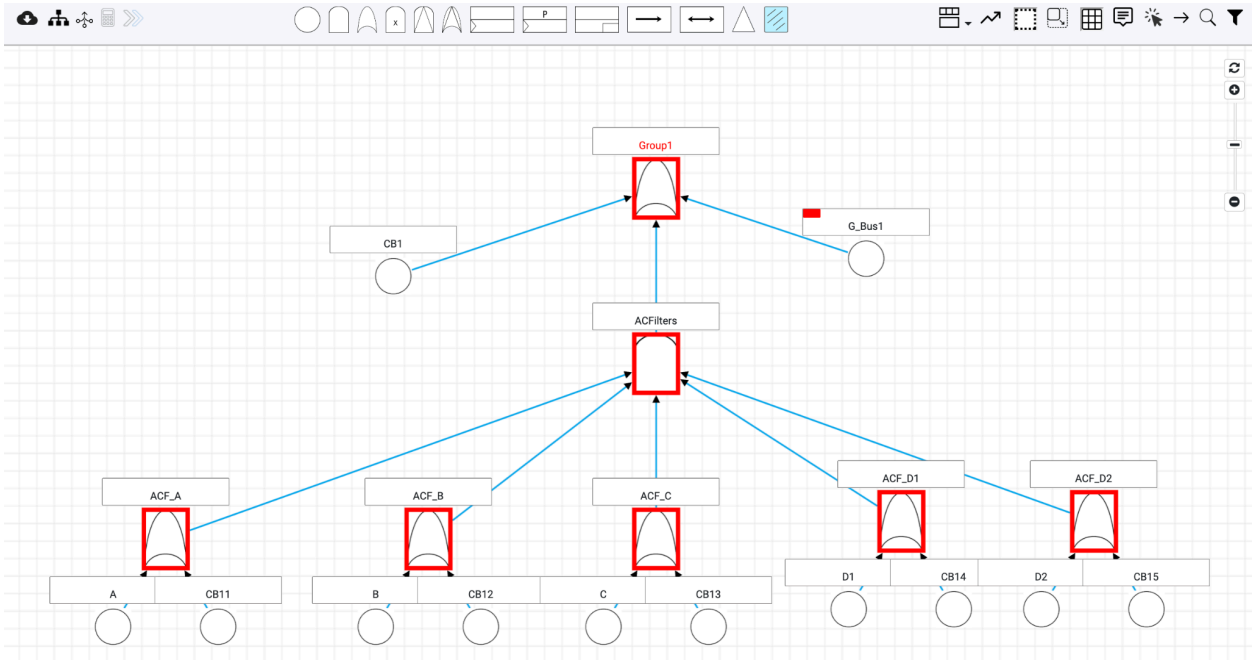
- A fault tree can be split on multiple pages, by generating mirrors of elements on different pages. This helps to improve the readability of fault trees.
- Only one element can be selected as a top-level element in a tree among all pages.
- The values of some of the fields e.g. failure rates, probabilities, etc can be given as parameters (defined later) instead of constant values.
- The “Generate label for the failure event” checkbox is only visible in the advance view.
- The “Generate label for the failure event” checkbox is available only for those elements that propagate failure to their parents. If this checkbox is selected for an element, it is added to the sub-section “Labelled Events” of the fault tree (in the left panel) as a basic event.
- Elements can be connected by drawing edges between them.
- A module can be annotated with attributes, reference manuals, and/or color tags.
- Click on the download icon  , a popup comes up to download the tree in JSON, Latex, and Galileo formats.


Export
Select file format
Galileo
A fault tree with a root element representing the root of the main page will be created and stored in a Galileo file.
Parameter set
HVDC_800KV_day
Before creating the fault tree, the model's parameters, if any, will be substituted with their values from the parameter set.
Galileo format will lose layout, reference manual, and attribute data.
Cancel
Export


Export
Select file format
Latex
A fault tree with a root element representing the root of the main page will be created and stored in a Latex file.
Parameter set
HVDC_800KV_day
Before creating the fault tree, the model's parameters, if any, will be substituted with their values from the parameter set.
Cancel
Export


Export
Select file format
JSON
All data given on the current page will be downloaded recursively.
The list of parameters, if any, used by the model will be created and saved in the JSON file together with the fault tree. Note that JSON format retains layout data whereas reference manual and attribute data might get lost. Moreover, it does not require a valid tree in order to export the data.
Cancel
Export

- Click on the icon  to highlight the elements that, along with their children, can be converted into modules (independent sub-trees). To convert an element and its children into a module, right-click on it and then click “Make Module”.



- To simplify a fault tree click on the down arrow along the icon  .

Simplification

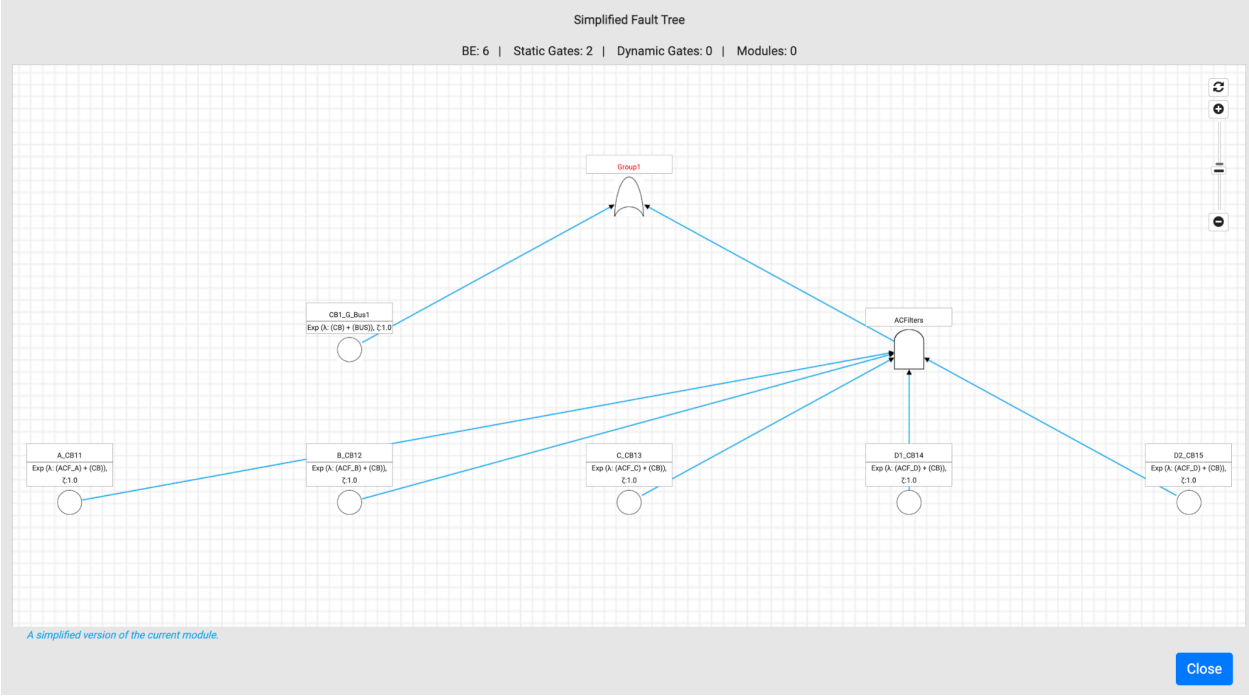
Apply	Rule	Abbreviation
☑	Split FDEPs with two or more children into single FDEPs with only one child.	SPLIT_FDEPS
☑	Try to merge BEs under an OR-gate into one BE.	MERGE_BES
☑	Trim parts of the DFT in place which do not contribute to the top level element.	TRIM
☐	Try to remove superfluous dependencies. These dependencies have a trigger which already leads to failure of the top level element.	REMOVE_DEPENDENCIES_TLE
☐	Try to merge gates with the same type and identical successors. These gates surely fail simultaneously and thus, one gate can be removed.	MERGE_IDENTICAL_GATES
☑	Remove gates with just one successor. These gates will fail together with this child, so they can directly be eliminated.	REMOVE_SINGLE_SUCCESSOR
☑	Flattening of AND-/OR-/PAND-gates.	FLATTEN_GATE
☐	Subsumption of OR-gate by AND-gate or of AND-gate by OR-gate.	SUBSUME_GATE
☐	Eliminate FDEPs by introducing an OR-gate. Let A be the trigger and B be the dependent element. Both must be connected to the top level element. B must have only one predecessor and no SPARE or PAND/POR in its predecessor closure.	REPLACE_FDEP_BY_OR
☐	Eliminate superfluous FDEP from AND or OR. This FDEP is triggered after the failure of the dependent element and thus, it does not influence anything else.	REMOVE_SUPERFLUOUS_FDEP
☐	Eliminate FDEP between two successors of an OR or PAND. Only supports FDEPs with one common predecessor.	REMOVE_SUPERFLUOUS_FDEP_SU...

☐ Replace modules with their corresponding sub-trees before simplifying the fault tree.

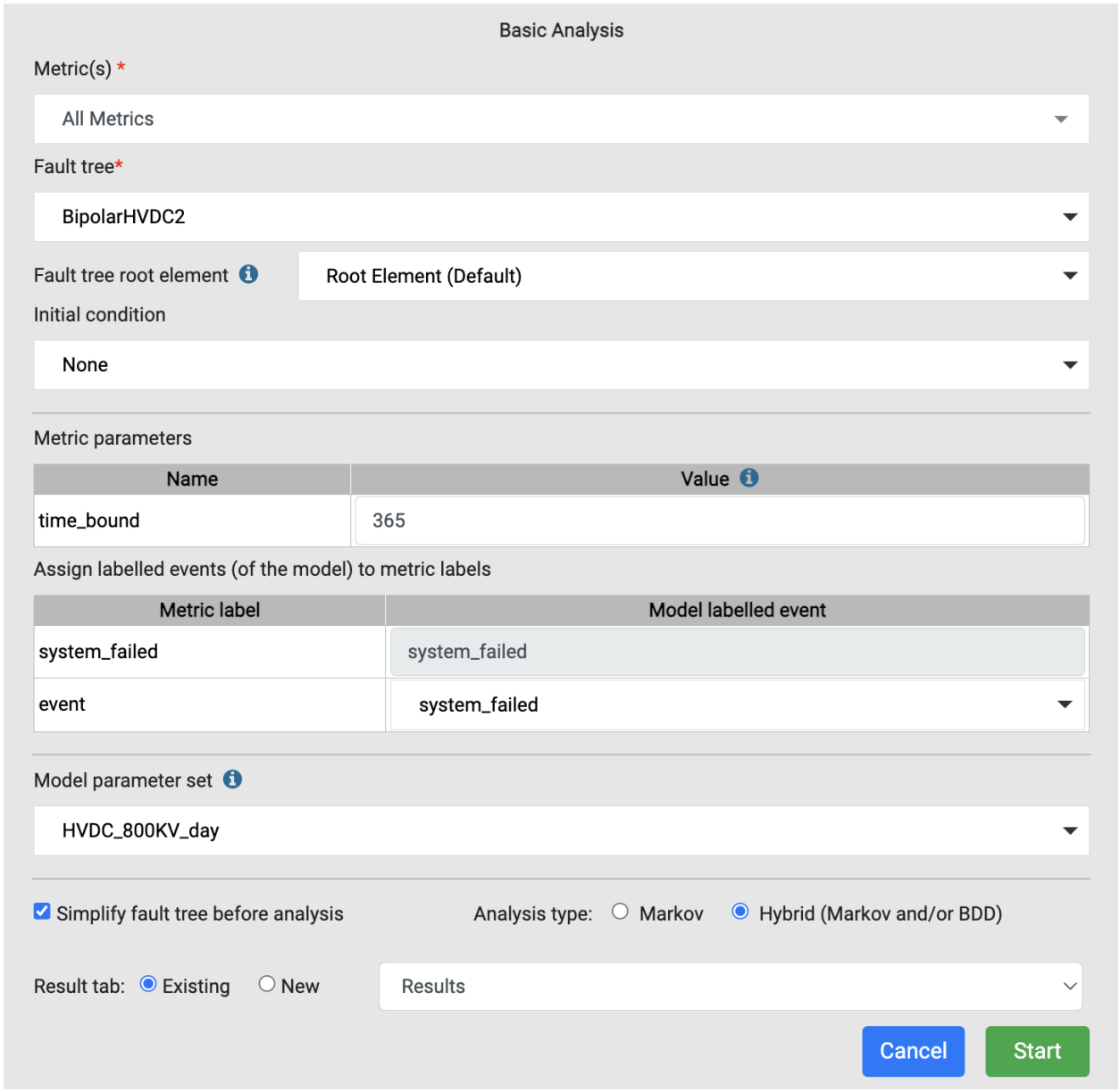
Simplification will only be applied to the fault tree given in the current module.

Cancel Simplify

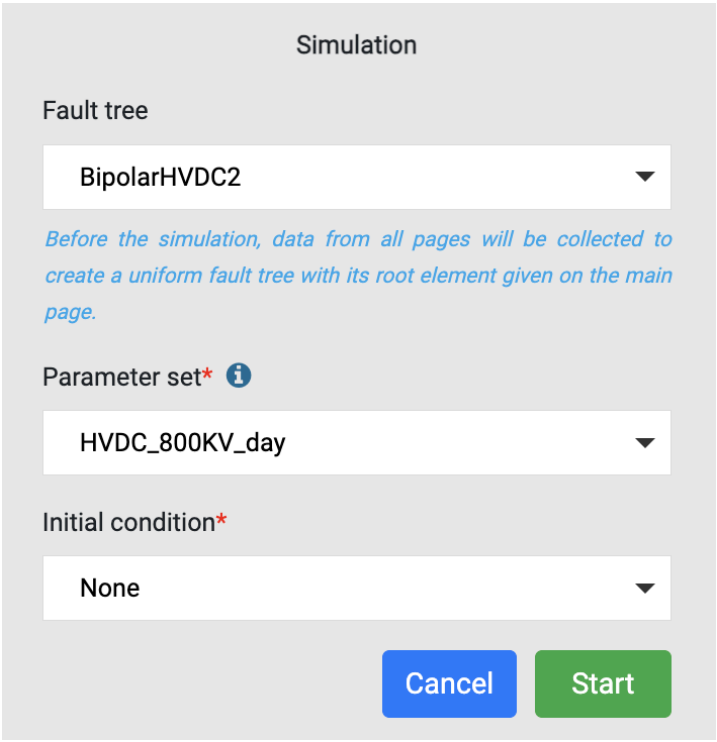
- Select rules that you want to apply for the simplification. The most important rules are selected by default.
- Check the radio button if you want to replace modules with their corresponding sub-trees before simplifying the fault tree.
- One can simplify a module or the whole fault tree spread across multiple pages.
- On clicking “Simplify”, a popup will appear with the simplified fault tree



- Click on the icon  (not visible when modules are in focus) to do a basic analysis that includes reliability, mean-time-to-failure, and average-failure-probability per hour. It will take the user to the “Analysis” window under “Computing” in the left panel.

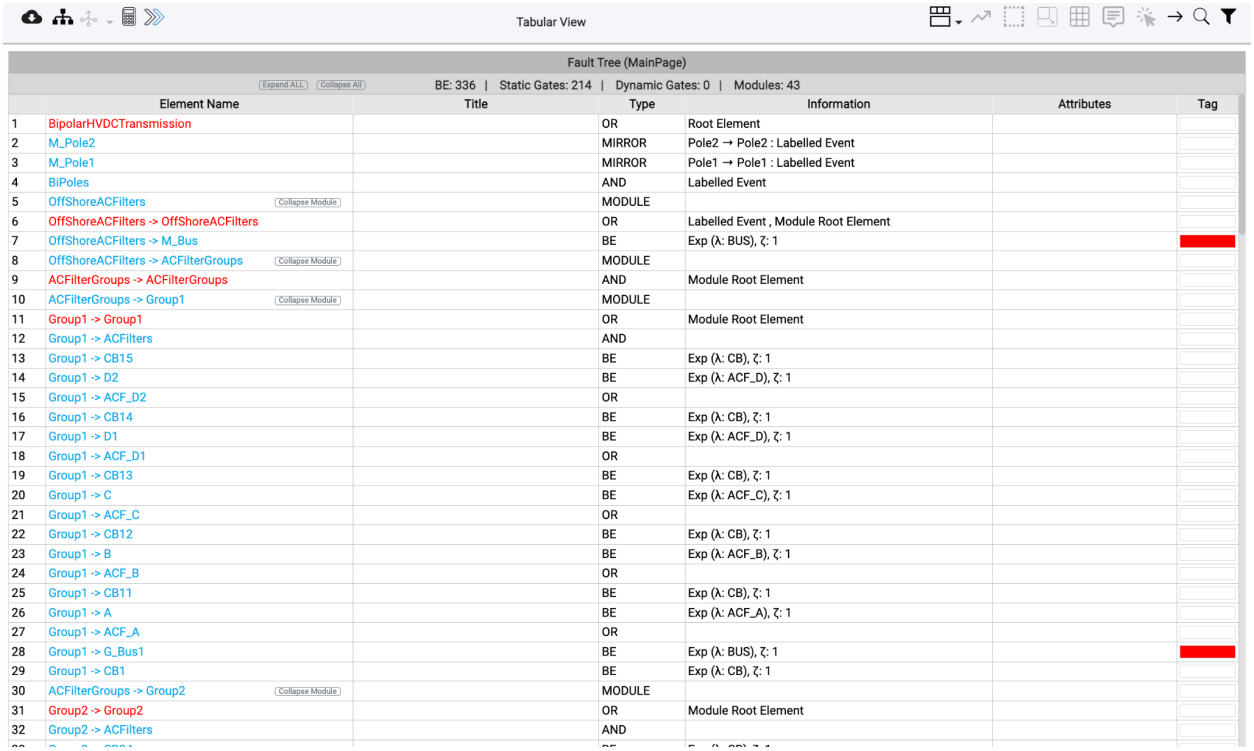


- Click on the icon  to start the interactive simulation. It will take the user to the "Interactive Simulation" window under "Computing" in the left panel. A popup will appear to select/change the model, a parameter set, and an Initial Condition. On clicking the "Start" button the simulation window appears.

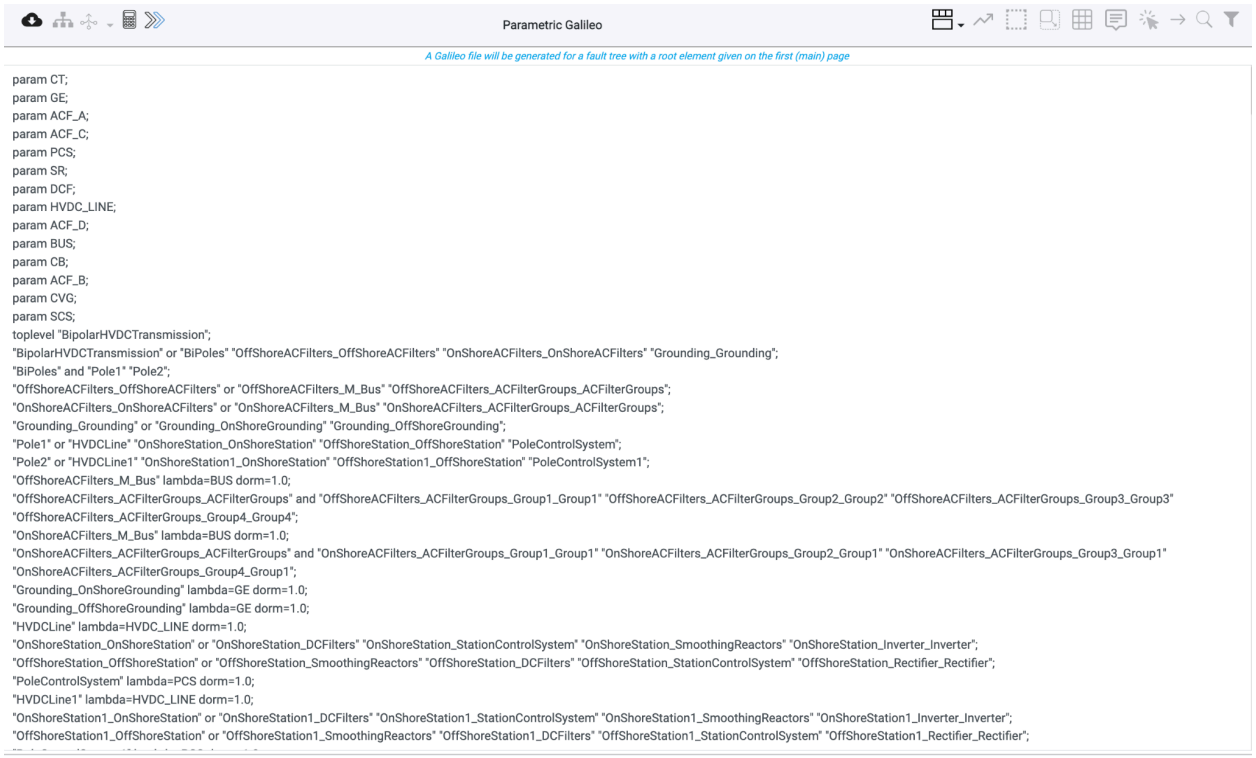


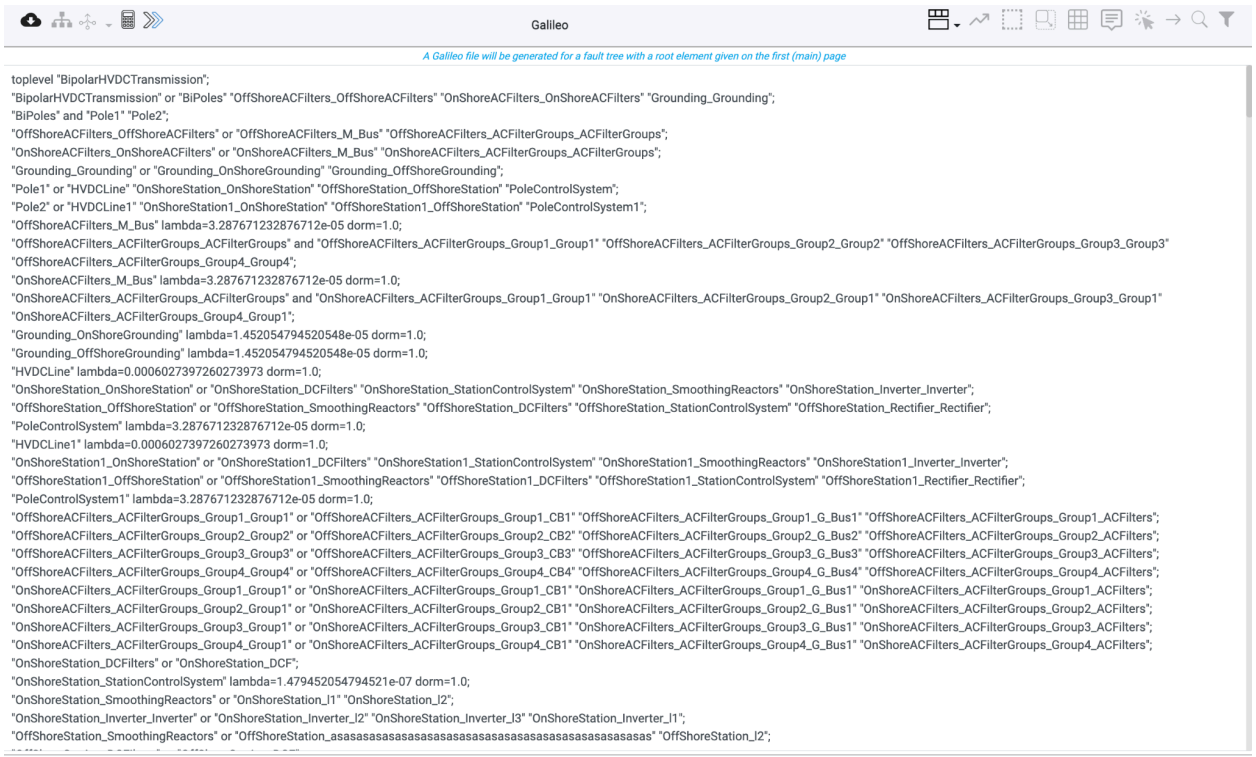
In case “Initial Condition” is applied, one fault tree is generated by collected data from all pages for simulation purposes. Otherwise, a simulation is run by keeping elements of each page separate from others.

- Click on the down arrow new the icon  , it will give four options to display a tree:
  - Canvas view. It shows the tree in the grid.
  - Tabular view. It will show the tree in a tabular form
  - Galileo view. It shows the tree in Galileo format.
  - Parametric Galileo view: It shows the tree in Galileo format along with the parameters of the model.

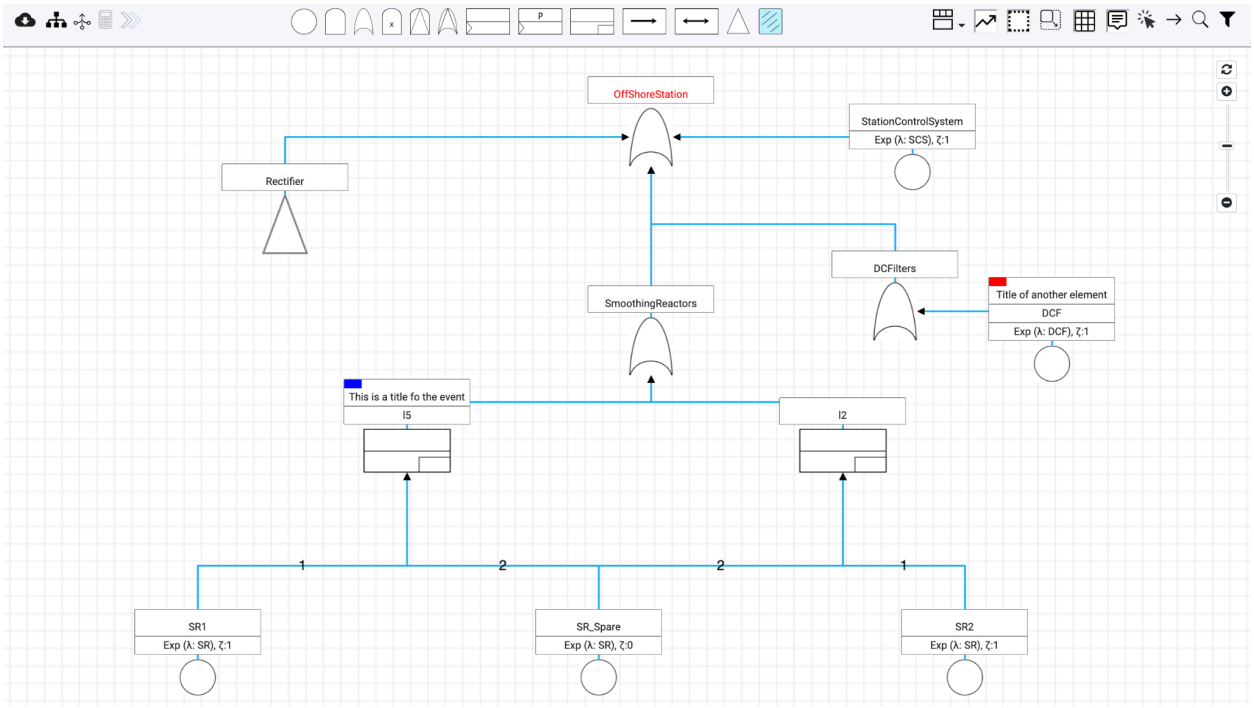


	Element Name	Title	Type	Information	Attributes	Tag
1	BipolarHVDCTransmission		OR	Root Element		
2	M_Pole2		MIRROR	Pole2 → Pole2 : Labelled Event		
3	M_Pole1		MIRROR	Pole1 → Pole1 : Labelled Event		
4	BiPoles		AND	Labelled Event		
5	OffShoreACFilters Collapse Module		MODULE			
6	OffShoreACFilters -> OffShoreACFilters		OR	Labelled Event , Module Root Element		
7	OffShoreACFilters -> M_Bus		BE	Exp (λ: BUS), ζ: 1		
8	OffShoreACFilters -> ACFilterGroups Collapse Module		MODULE			
9	ACFilterGroups -> ACFilterGroups		AND	Module Root Element		
10	ACFilterGroups -> Group1 Collapse Module		MODULE			
11	Group1 -> Group1		OR	Module Root Element		
12	Group1 -> ACFilters		AND			
13	Group1 -> CB15		BE	Exp (λ: CB), ζ: 1		
14	Group1 -> D2		BE	Exp (λ: ACF_D), ζ: 1		
15	Group1 -> ACF_D2		OR			
16	Group1 -> CB14		BE	Exp (λ: CB), ζ: 1		
17	Group1 -> D1		BE	Exp (λ: ACF_D), ζ: 1		
18	Group1 -> ACF_D1		OR			
19	Group1 -> CB13		BE	Exp (λ: CB), ζ: 1		
20	Group1 -> C		BE	Exp (λ: ACF_C), ζ: 1		
21	Group1 -> ACF_C		OR			
22	Group1 -> CB12		BE	Exp (λ: CB), ζ: 1		
23	Group1 -> B		BE	Exp (λ: ACF_B), ζ: 1		
24	Group1 -> ACF_B		OR			
25	Group1 -> CB11		BE	Exp (λ: CB), ζ: 1		
26	Group1 -> A		BE	Exp (λ: ACF_A), ζ: 1		
27	Group1 -> ACF_A		OR			
28	Group1 -> G_Bus1		BE	Exp (λ: BUS), ζ: 1		
29	Group1 -> CB1		BE	Exp (λ: CB), ζ: 1		
30	ACFilterGroups -> Group2 Collapse Module		MODULE			
31	Group2 -> Group2		OR	Module Root Element		
32	Group2 -> ACFilters		AND			



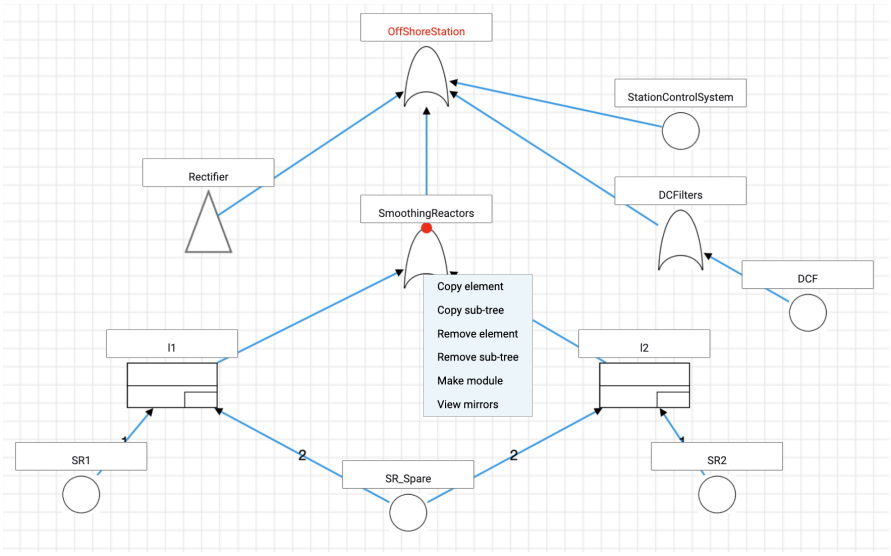


- Click on  icon to change the type of edges from curved to straight.



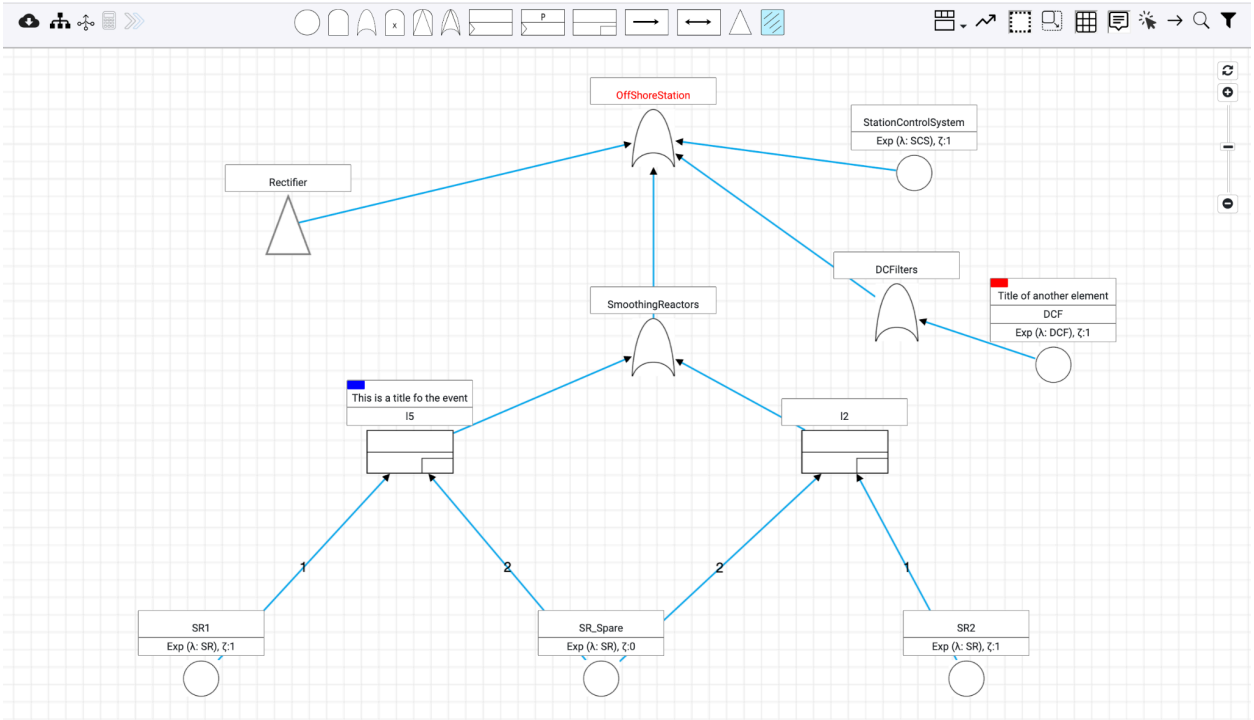
- Click on the icon  to enable the selection of multiple elements in the grid view. This can be done by clicking on a screen and then drawing the mouse. All elements with a rectangle will be selected, which can then be moved together.
- Right-click on the canvas and the following popup will appear:

  - By clicking “Remove Elements” all selected elements will be removed.
  - By clicking “Copy elements” all selected elements will be copied.
  - By clicking “Paste” previously copied elements will be pasted.
- On right-clicking on an element, a popup comes up that allows you to copy the element, copy the sub-tree under it, delete the element, delete the sub-tree under it, or convert the sub-tree under it into a module (if possible), and view mirrors of the selected element.

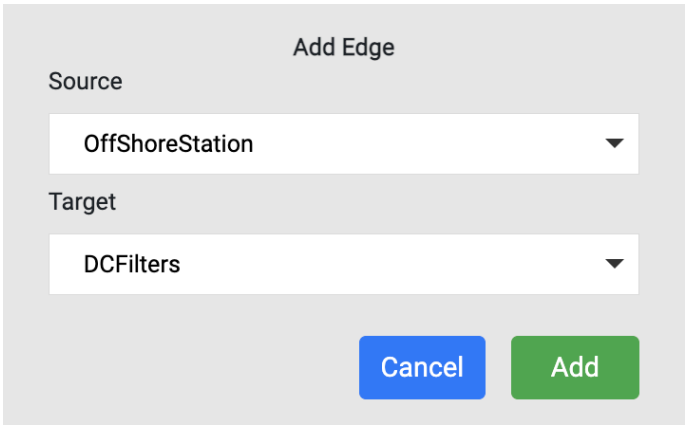


- Right-click an element and then select “View Mirrors”, a popup window will come up showing all mirrors of the selected element.

- Click  icon to enable the navigator at the bottom of the screen.
- Click  icon to show the grid on the screen.
- Click  icon to show the summary information about each element on the screen.

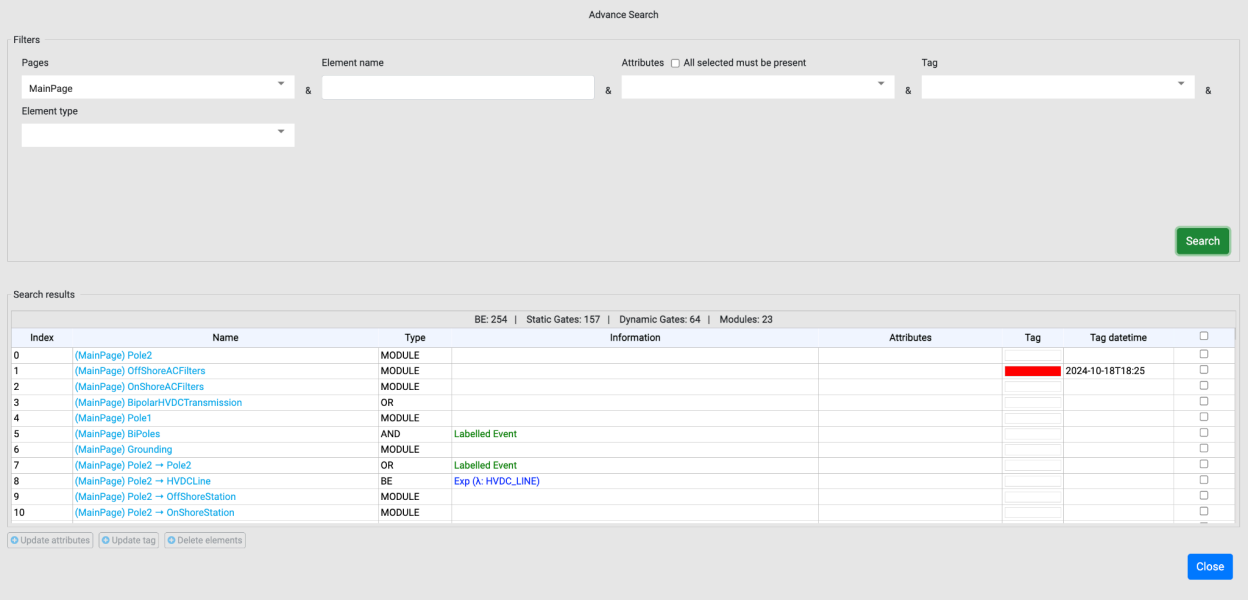


- Click  icon to display summary information about an element on hovering.
- Click  icon to draw edges between two elements of a fault tree. Both elements must be on the same page/module.

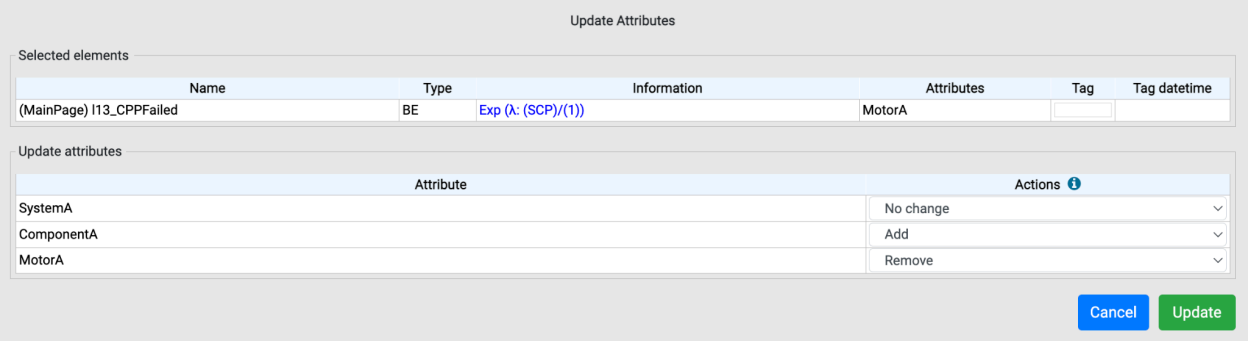


- Click  icon to search for any element on the screen.
- Click  icon to search elements based on filters. A popup will appear to apply filters.
    - All fields that are filled up, their data is AND together to find the nodes. Within a field, the data is OR-together unless it is mentioned otherwise.
    - If any file is not filled up, it is not considered to be a part of the filtering process, and thus ignored.

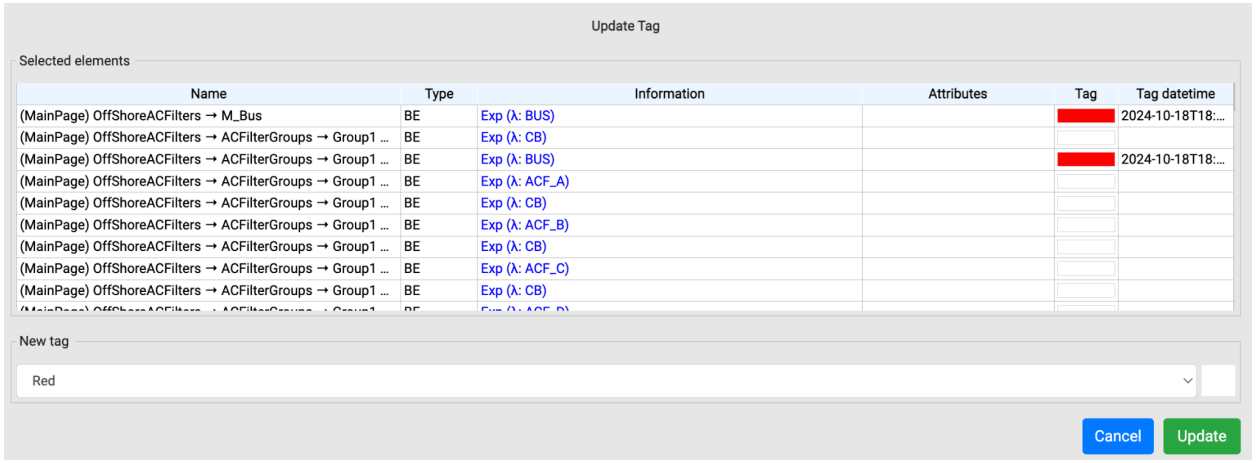
- On clicking the “Search” button, the result will be displayed in the “Search results” table.



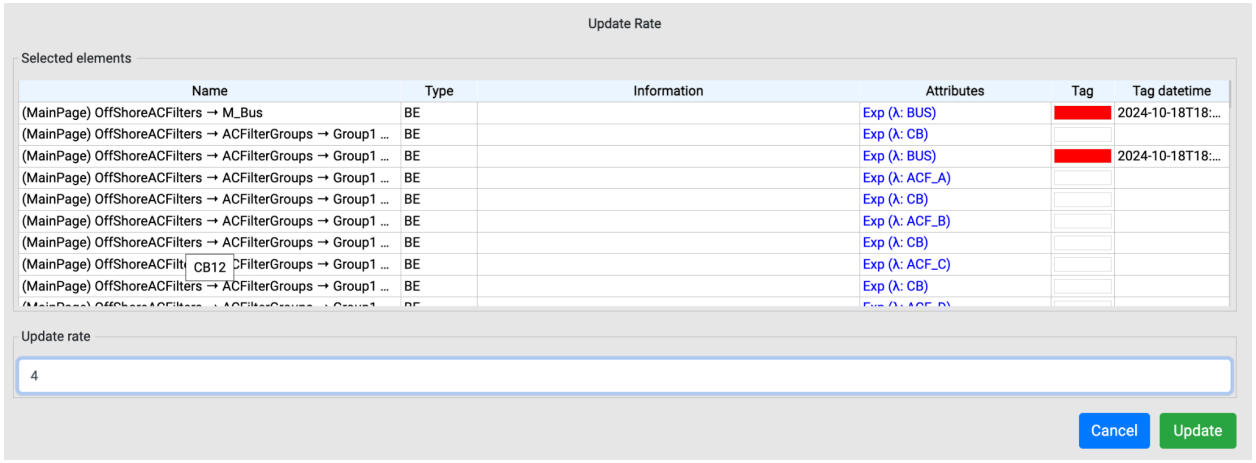
- An element can be updated by clicking its name and opening the corresponding popup.
- By selecting elements of the same type in the “Search results” table, operations can be performed on them at once. For example, in the case of BEs, the following operations can be performed:
  - Update attributes



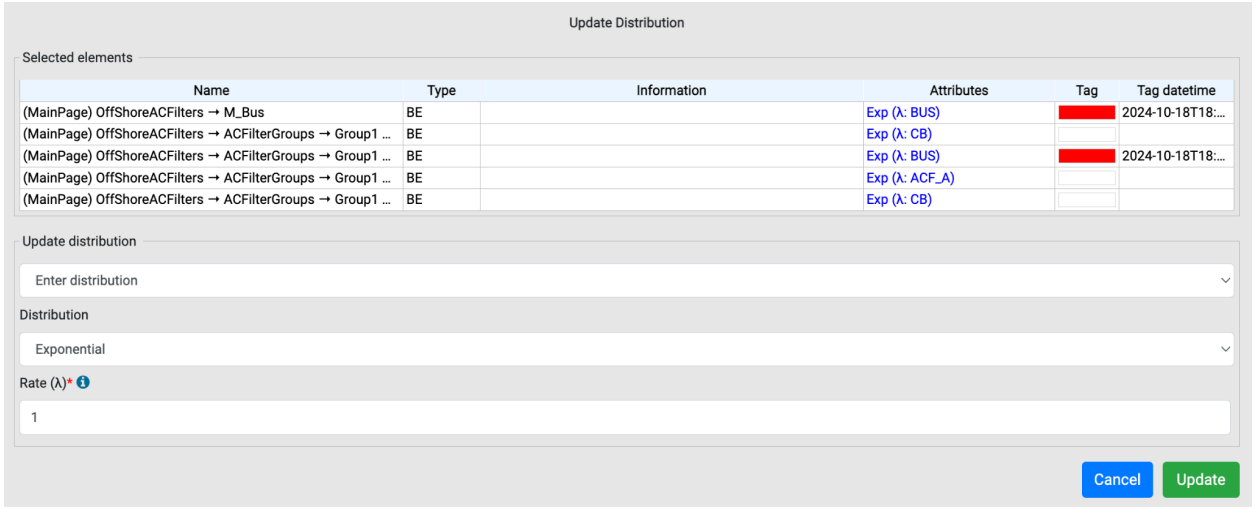
  - Update tags



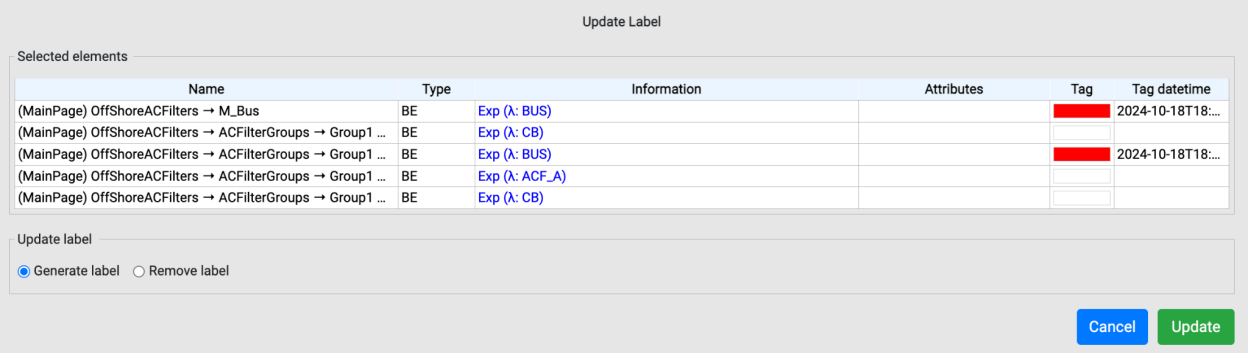
- Delete elements
- Update rates



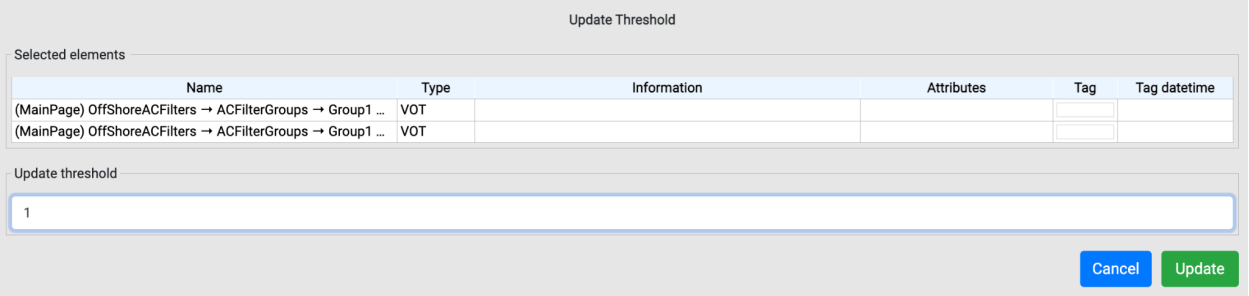
- Update distribution



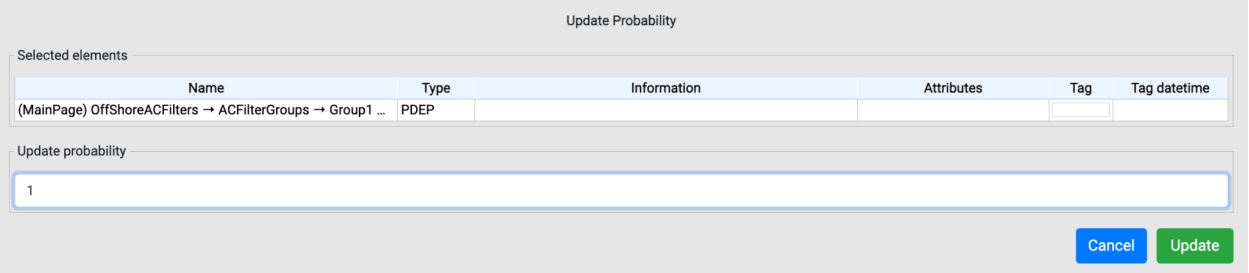
- Update label



- The threshold values of multiple VOT gates can be changed at once.



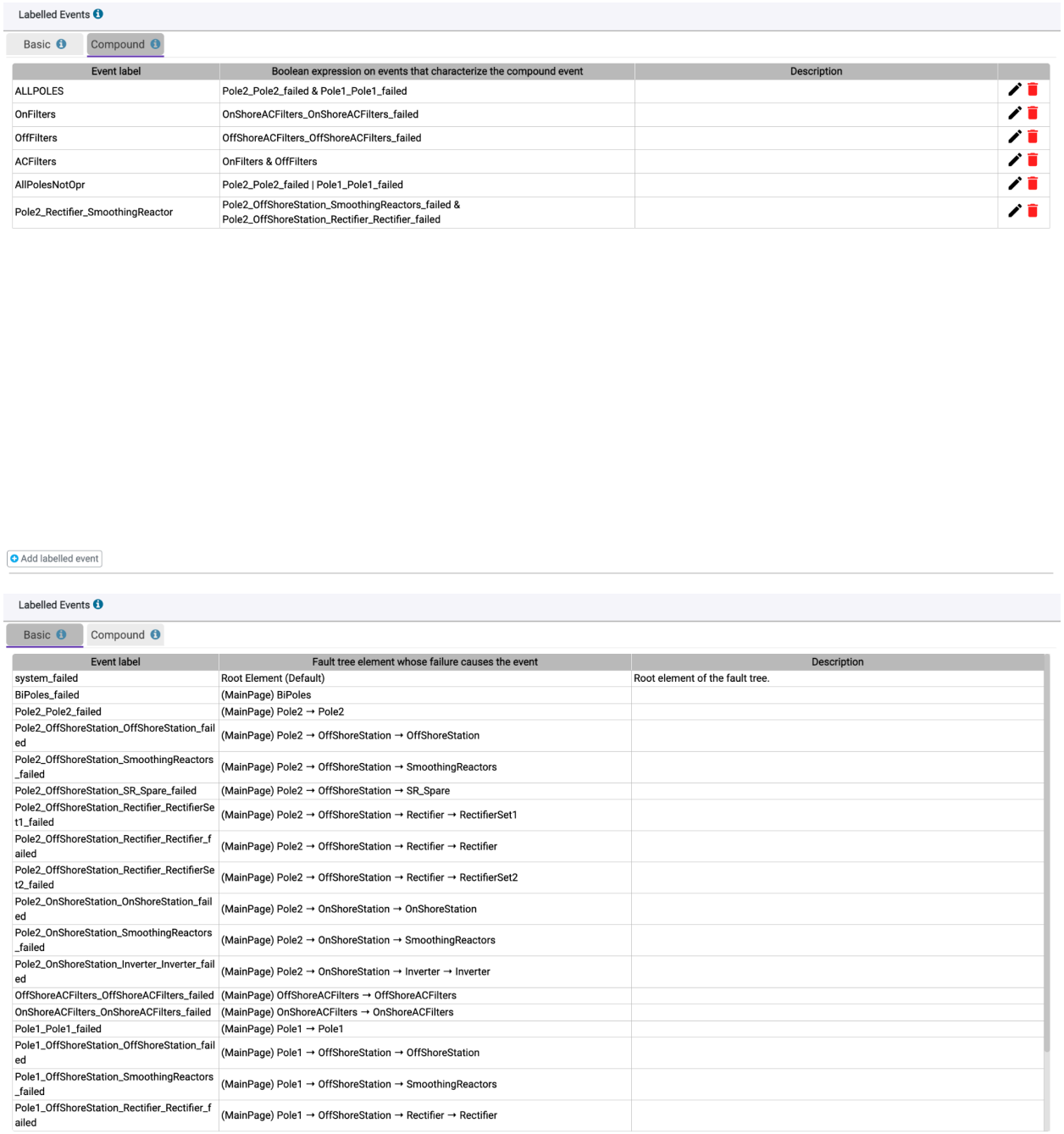
- The probability field of multiple PDEP gates can be changed at once.



Labelled Events

All fault propagating elements whose checkbox “Generate label for the failure event” is selected, appear as basic labelled events on the Labelled Events page. Compound events can be defined by boolean equations on (basic and compound) labelled events. Advanced measures can be computed on basic as well as compound labelled events.

In the advanced view, each fault tree has a section “Labelled Events” separated into Basic and Compound events.



Labelled Events

Basic | Compound

Event label	Boolean expression on events that characterize the compound event	Description
ALLPOLES	Pole2_Pole2_failed & Pole1_Pole1_failed	
OnFilters	OnShoreACFilters_OnShoreACFilters_failed	
OffFilters	OffShoreACFilters_OffShoreACFilters_failed	
ACFilters	OnFilters & OffFilters	
AllPolesNotOpr	Pole2_Pole2_failed | Pole1_Pole1_failed	
Pole2_Rectifier_SmoothingReactor	Pole2_OffShoreStation_SmoothingReactors_failed & Pole2_OffShoreStation_Rectifier_Rectifier_failed	

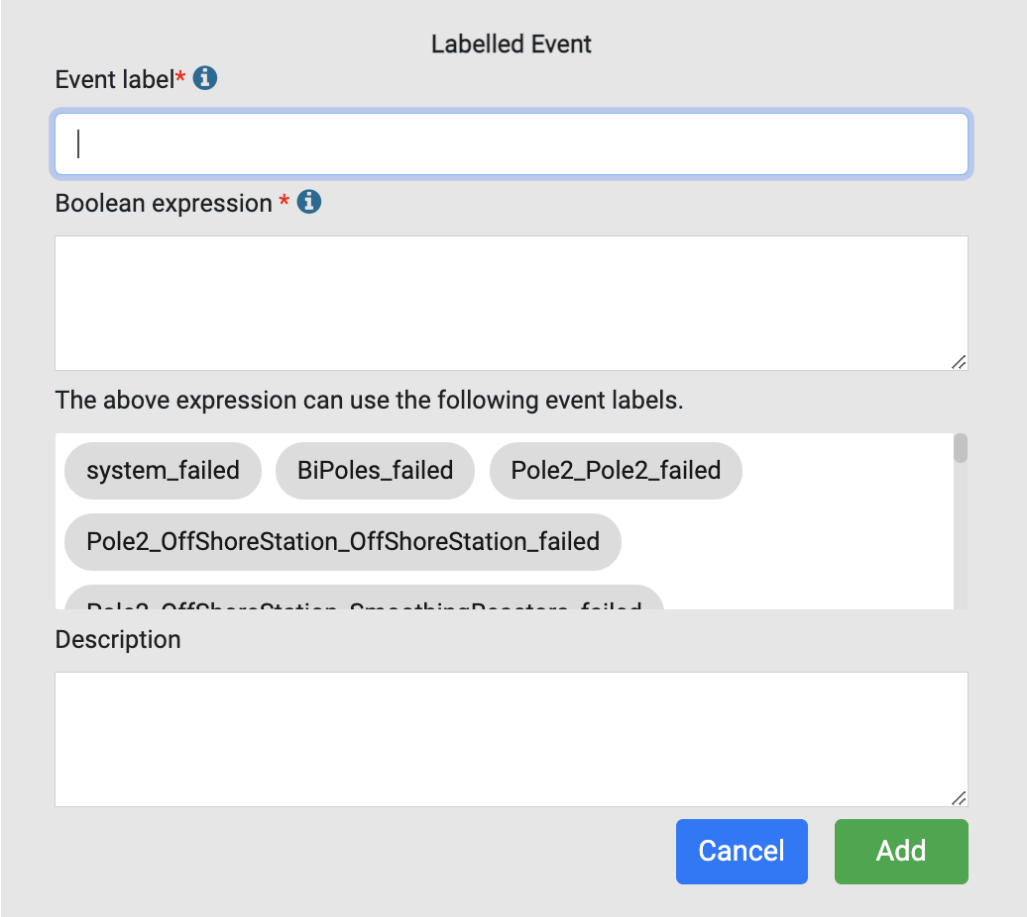
Add labelled event

Labelled Events

Basic | Compound

Event label	Fault tree element whose failure causes the event	Description
system_failed	Root Element (Default)	Root element of the fault tree.
BiPoles_failed	(MainPage) BiPoles	
Pole2_Pole2_failed	(MainPage) Pole2 → Pole2	
Pole2_OffShoreStation_OffShoreStation_failed	(MainPage) Pole2 → OffShoreStation → OffShoreStation	
Pole2_OffShoreStation_SmoothingReactors_failed	(MainPage) Pole2 → OffShoreStation → SmoothingReactors	
Pole2_OffShoreStation_SR_Spare_failed	(MainPage) Pole2 → OffShoreStation → SR_Spare	
Pole2_OffShoreStation_Rectifier_RectifierSet1_failed	(MainPage) Pole2 → OffShoreStation → Rectifier → RectifierSet1	
Pole2_OffShoreStation_Rectifier_Rectifier_failed	(MainPage) Pole2 → OffShoreStation → Rectifier → Rectifier	
Pole2_OffShoreStation_Rectifier_RectifierSet2_failed	(MainPage) Pole2 → OffShoreStation → Rectifier → RectifierSet2	
Pole2_OnShoreStation_OnShoreStation_failed	(MainPage) Pole2 → OnShoreStation → OnShoreStation	
Pole2_OnShoreStation_SmoothingReactors_failed	(MainPage) Pole2 → OnShoreStation → SmoothingReactors	
Pole2_OnShoreStation_Inverter_Inverter_failed	(MainPage) Pole2 → OnShoreStation → Inverter → Inverter	
OffShoreACFilters_OffShoreACFilters_failed	(MainPage) OffShoreACFilters → OffShoreACFilters	
OnShoreACFilters_OnShoreACFilters_failed	(MainPage) OnShoreACFilters → OnShoreACFilters	
Pole1_Pole1_failed	(MainPage) Pole1 → Pole1	
Pole1_OffShoreStation_OffShoreStation_failed	(MainPage) Pole1 → OffShoreStation → OffShoreStation	
Pole1_OffShoreStation_SmoothingReactors_failed	(MainPage) Pole1 → OffShoreStation → SmoothingReactors	
Pole1_OffShoreStation_Rectifier_Rectifier_failed	(MainPage) Pole1 → OffShoreStation → Rectifier → Rectifier	

Click on the “Add labelled event” to create a new compound event.



Enter the event label and boolean expression on existing events shown below in the field “The above expression can use the following Event labels”. On filling in mandatory fields, click the “Add” button.

CCF Groups

Common cause failures (CCF): a sub-category of dependent failures, representing events where multiple failures occur due to a shared cause. They are important to consider because they can violate the effects of redundancy. CCFs may violate the performance of an individual safety barrier, or result in the simultaneous failure of several safety barriers. Semi-automatic common cause failure (CCF) modeling can be performed by placing basic elements into CCF groups. This approach to modeling CCFs is more practical than the typical modeling method, which involves adding CCF events by hand to the fault trees. When modeling systems with high levels of redundancy, in particular, such CCF modeling in the fault trees can occasionally greatly increase the overall size and complexity of the fault trees.

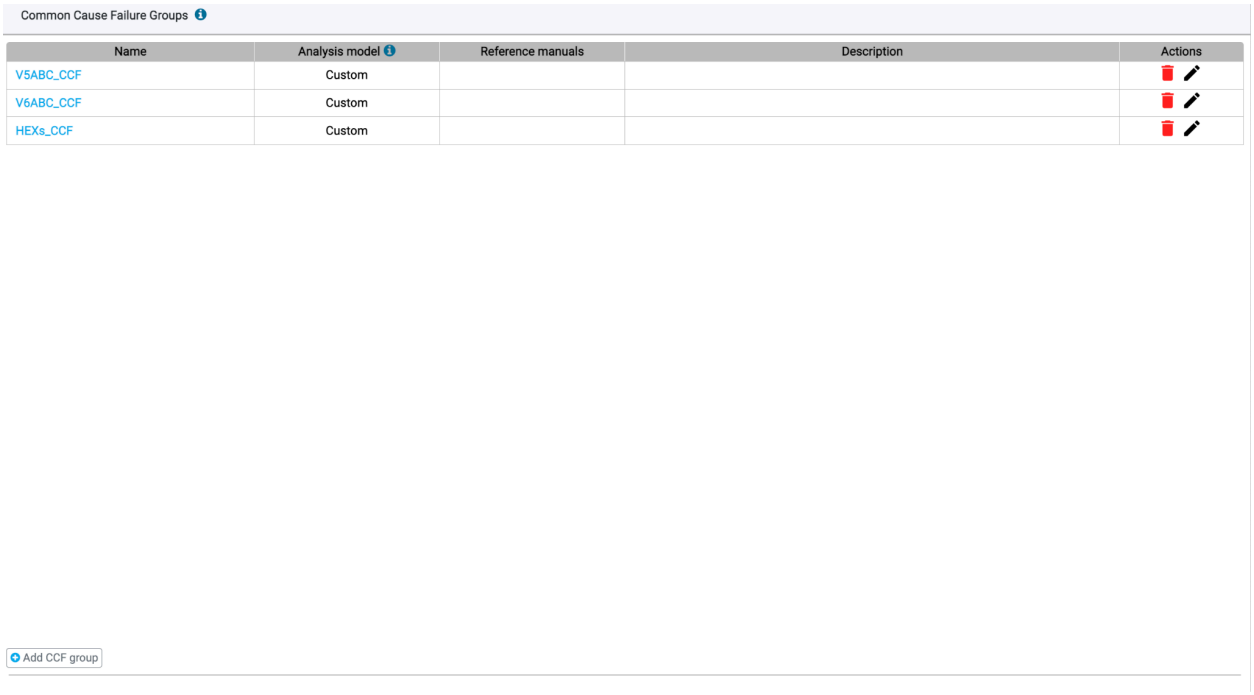
CCF Elements & CCF Events: CCF elements are a set of basic elements of a fault tree that (may) fail due to certain common causes. Whereas CCF events are the causes of failures of CCF elements. CCF groupings are groups made up of these CCF elements.

A fault tree structure (CCF Tree), which is an OR-gate that takes in CCF events (causes of failures of CCF elements) as inputs, is automatically generated for each element in a CCF group to illustrate the propagation of failure due to common causes. Each CCF element in a CCF group is substituted with its matching CCF tree in the original fault tree before analysis. Assume that a fault tree has four redundant components represented by the basic elements A, B, C, and D, placed in a CCF group. The following CCF events are automatically created by the SAFEST for this group: AB, AC, AD, BC, BD, CD, ABC, ABD, ACD, BCD, and ABCD; for example, AB denotes a failure event involving components A and B jointly. Subsequently, the software generates CCF trees for CCF elements. Before analysis, these CCF trees will replace the CCF elements in CCF groups in the fault tree. For instance, the program generates a CCF tree for the basic element A, with an OR-gate acting as the top gate and events A (individual failure), AB, AC, AD, ABC, ABD, ACD, and ABCD as inputs.

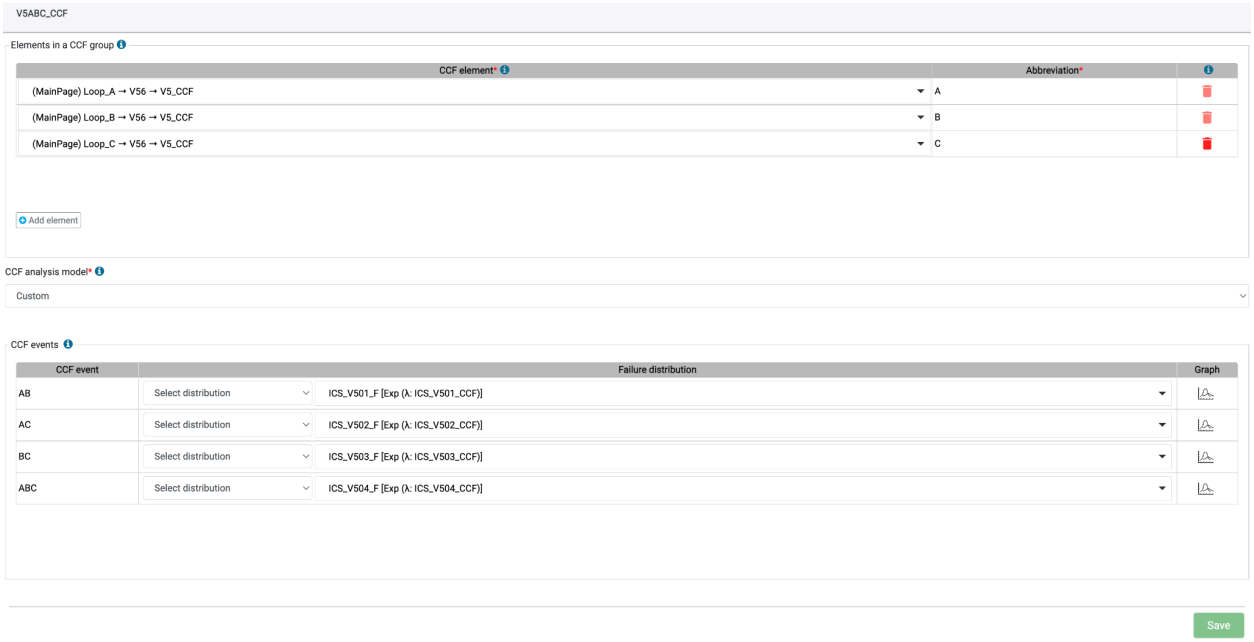
Analysis Models: There are two types of CCF modeling: explicit and implicit. Explicit CCF modeling is used once each distinct cause of CCF (CCF event) is identified, along with its reliability data e.g. probability of failure. Human error, shared equipment, utility problems (such as power, cooling, heating failure, or loss of hydraulic power), and natural events (such as lightning, flooding, or storms) are some of the specific causes.

Implicit modeling may be used when the reliability data of CCF events are unknown. Their failure probabilities are quantified using various analysis models. These models aid in determining the proportion of component failures attributable to common causes. There are several implicit models, e.g. Beta-factor model, the Alpha-factor model, the Multiple Greek Letter model, the Binomial failure rate model, etc. Each model has its own set of parameters required to quantify probabilities of CCF events. In the case of the Custom Model, reliability data for each CCF event is defined manually.

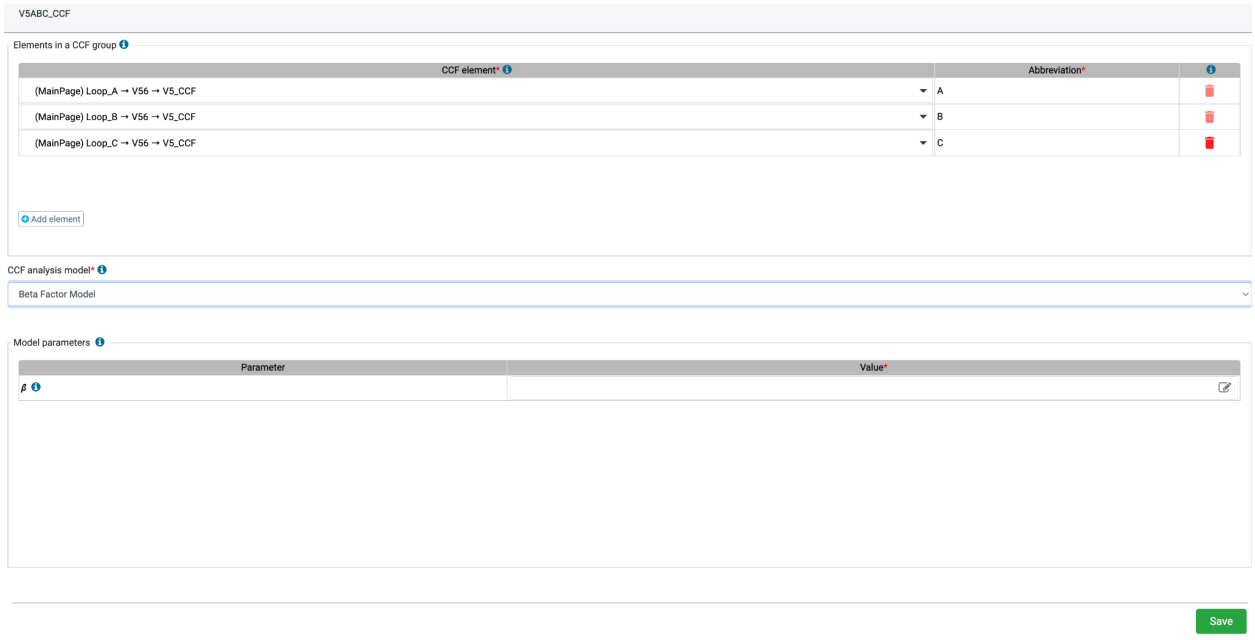
For a fault tree, we can create common cause failures (CCF) groups, which are displayed on the “Common Cause Failure Groups” page under a fault tree. These CCF groups may be incorporated into a fault tree before analysis.



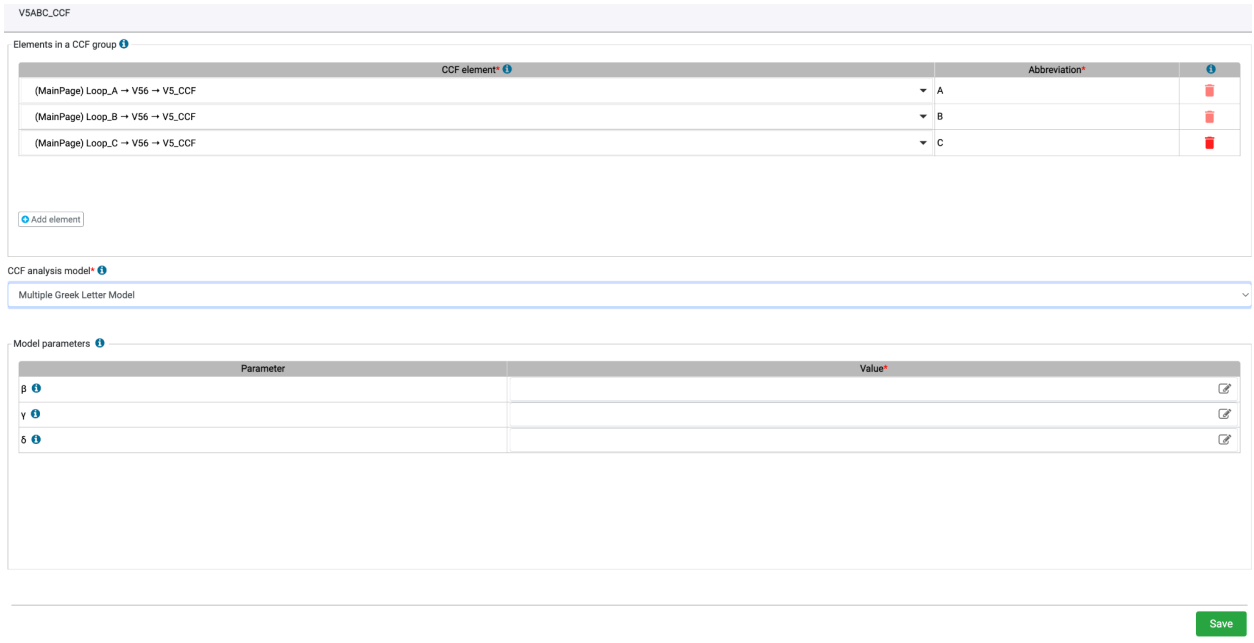
- Click the “Add CCF group” button to create a new group.



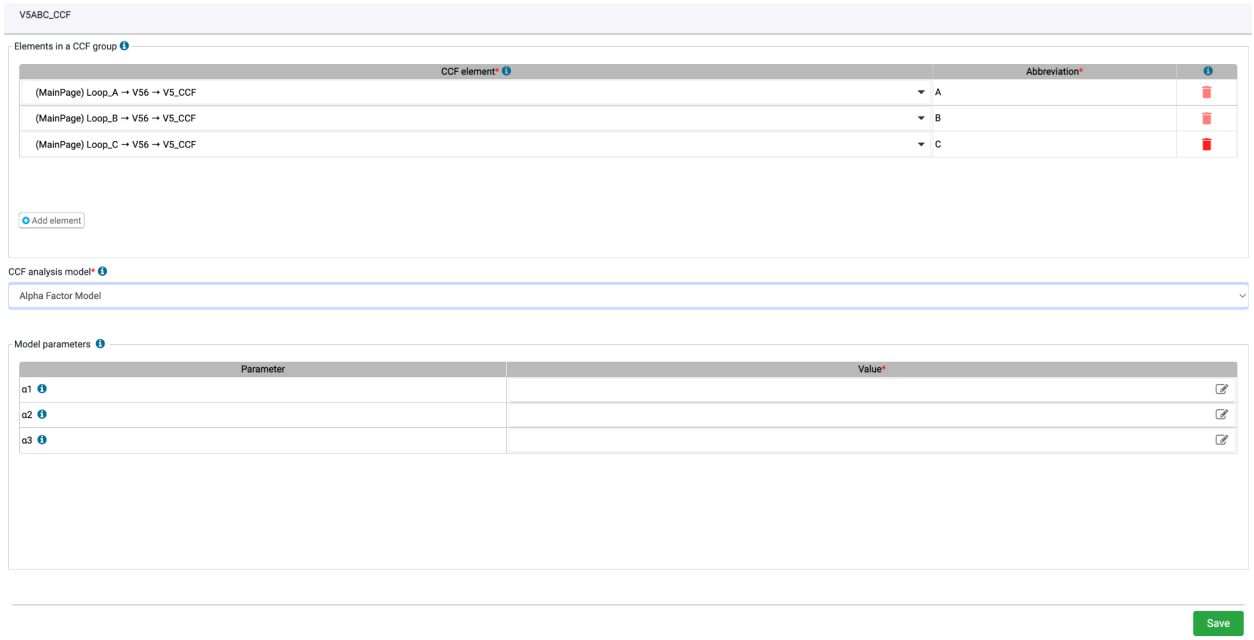
- Add fault tree elements (CCF elements) to the group. Only BEs can be added to CCF elements in a group.
- Select “CCF analysis model”. For each model, there is a set of parameters that need to be defined. Currently, the following analysis models are supported by SAFEST.
    - Beta Factor Model



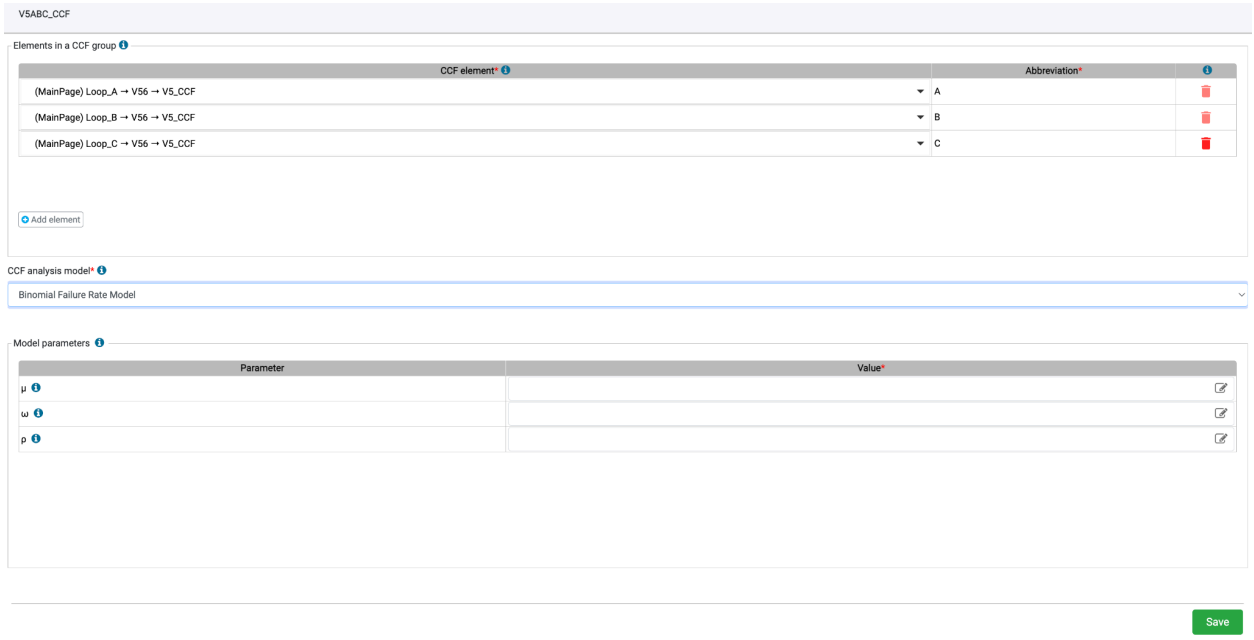
- Multiple Greek Letter Model



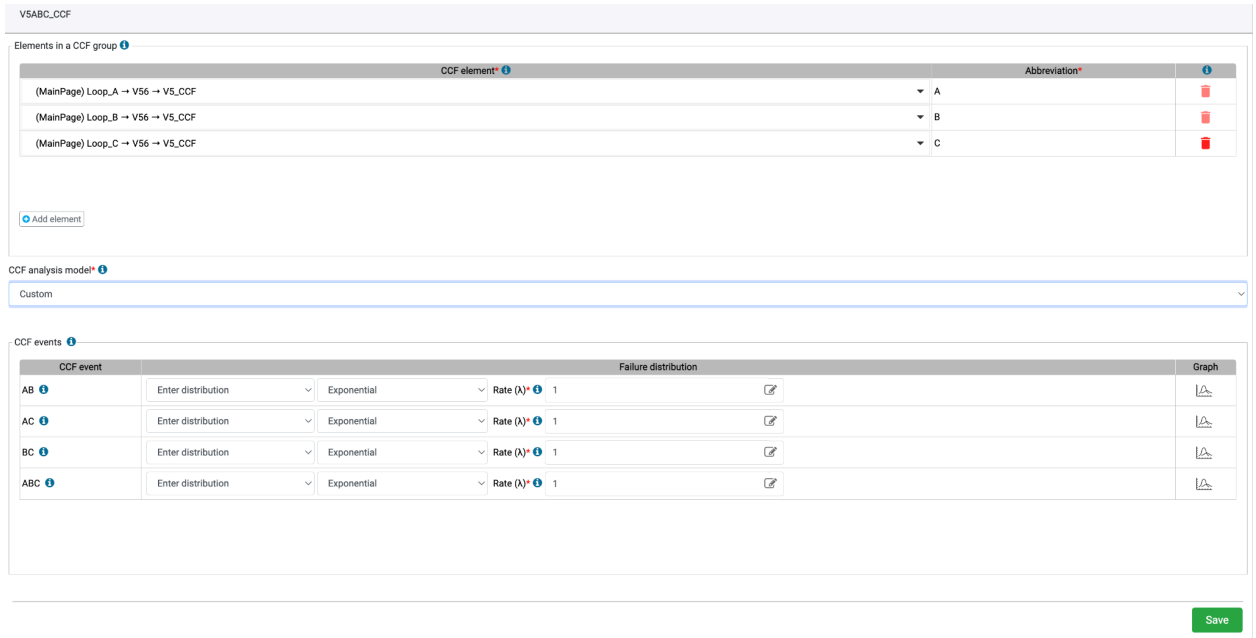
- Alpha factor Model



- Binomial Model



- Custom Model
  - In the case of the custom model, the failure distribution of each CCF event has to be defined.

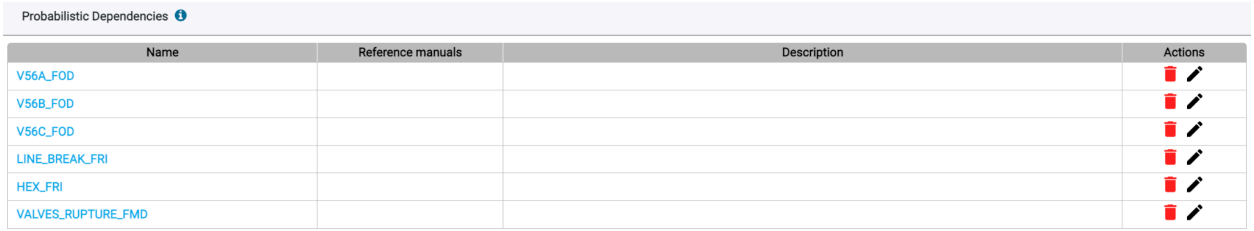


Probabilistic Dependencies

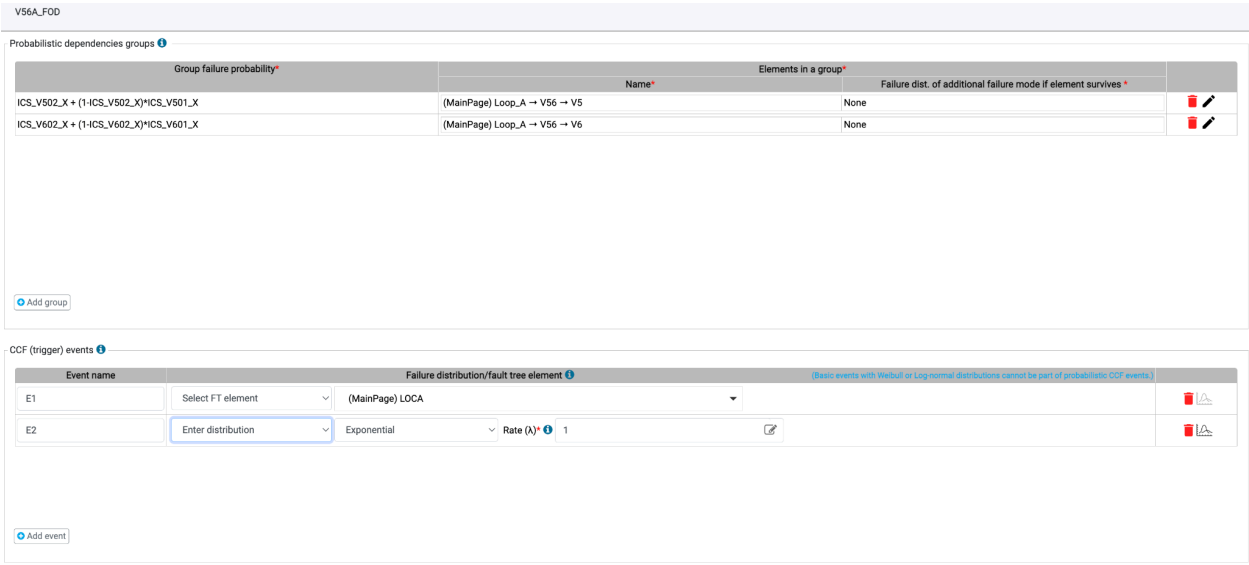
Probabilistic functional dependencies may exist in a system, e.g., failure-on-demand, failure-rate-increment, or addition of a new failure-mode in case an element services the impact of an (accidental) event, etc. Instead of modeling them explicitly in fault trees, SAFEST allows for specifying them separately in a tabular form. These dependencies are applied to a fault tree before analysis. This helps simplify the modeling process enormously.

For a fault tree, we can create probabilistic dependencies groups, which are displayed on the “Probabilistic Dependencies” page under a fault tree. All elements in a probabilistic CCF group may fail with a given probability when a common cause (CC) of failure occurs. As CCF elements in a group may not fail with a positive probability, their failure rates might be increased or/and a new failure mode can be active for them, thus a new fault tree emerges in a post-CCF event scenario.

Before analysis, for each CCF group, say G, a dynamic fault tree structure is constructed. For each element, say A, in group G, a new OR-gate is created that accepts two inputs, the element A and a new basic element, say B, that models e.g. an incremented failure rate for the element A. The element B is made the second child of a new SEQ gate, and element A is made the dependent child of a new PDEP gate that models the failure probability of group G on the occurrence of common causes. Both PDEP and SEQ gates have their first child another OR-gate that has all CCF (trigger) events as its children.



- Click the “Add probabilistic dependency” button to create a new prob. Dependency.



- Each CCF group in the “Probabilistic Dependencies Groups” table is affected if any event mentioned in the “CCF (trigger) events” table occurs.
- On the occurrence of a CCF (trigger) event, elements in a CCF group fail with a probability given in “Group failure probability”.
    - In case the group failure probability is less than one, the CCF elements can survive. In this case, one can specify the failure distribution of additional failure mode in the “Failure dist. Of additional failure mode if element services” column.

- Click the “Add event” button to create a new CCF event in the “CCF (trigger) events” table. Failure distribution of the new CCF event can be
    - Specified manually, or
    - Derived from an element of the fault tree

Initial Conditions

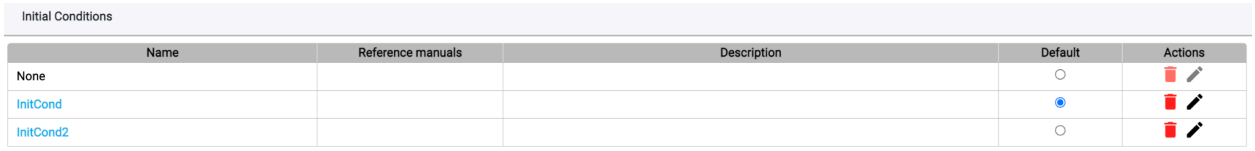
Initial conditions/boundary conditions consist of:

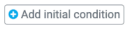
- CCF groups,
- Probabilistic dependencies, and
- Evidence

that we apply before analyzing a fault tree.

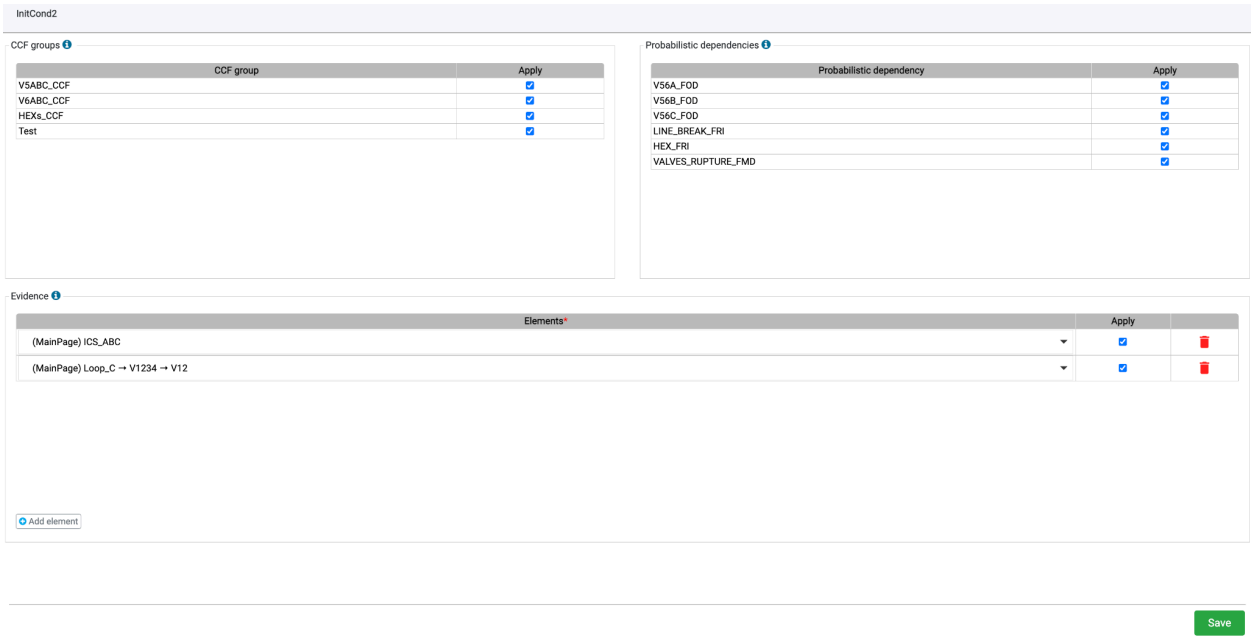
Evidence is a group of modules that are assumed to have already failed. A module is a subtree with a root node, say v, if all failures within the subtree have to propagate through the root node v. Thus, the state of the entire module is represented by the state of its root node. We call a module a dynamic module if it contains at least one dynamic gate; otherwise, it is a static module. During analysis, all possible failure orderings (traces) of the modules in the evidence are examined. Note that the elements within a module cannot be selected as evidence because their failure ordering might impact the failure behavior of its corresponding module.

For a fault tree, we can create Initial Conditions, which are displayed on the “Initial Conditions” page under a fault tree. One can apply any initial condition for analyzing the fault tree.





- Click the “Add initial condition” button to create a new condition.



- The page contains all CCF groups and probabilistic dependencies that we have already created for the fault tree.
- Select “Apply” checkboxes for the CCF groups and prob. Dependencies that we want to apply to the fault tree before analysis.
- Click the “Add element” button in the “Evidence” table to add elements of the fault trees that we assume have already failed at the start of the analysis as evidence of failure.
- Select “Apply” checkboxes for the elements in the “Evidence” table that we assume failed at the start of the analysis.

Configurations

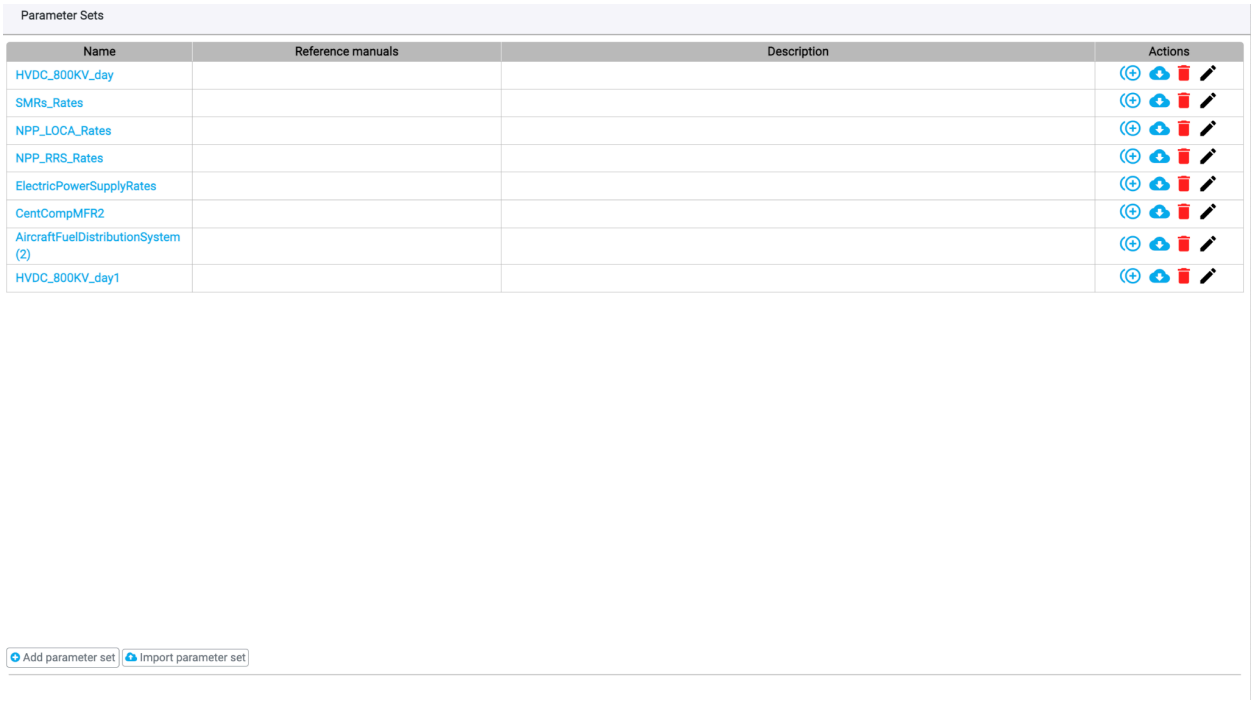
Parameter Sets

Each parameter set contains a list of parameters that are key-value pairs. They are used to specify values of e.g. probabilities, failure rates, etc. in fault tree models. By changing their values several variants of fault trees can be generated, which can then be compared with each other based on the results of metrics of interest. Each parameter set comprises:

- Constants
- Constant expressions
- Failure event distributions (Exponential, Erlang, Weibull, Log-normal)
- Failure event empirical distributions – failure distributions generated from data sets.

View

Click the “Parameter Sets” under “Configurations” in the left panel to view all existing parameter sets.



Creation

Click the “Add parameter set” button to create a new parameter.

Import

Click the “Import parameter set” to import the parameter set in .ps format (a format in which parameter sets are imported/exported in SAFEST).

Duplicate

Click the icon to duplicate a parameter set.

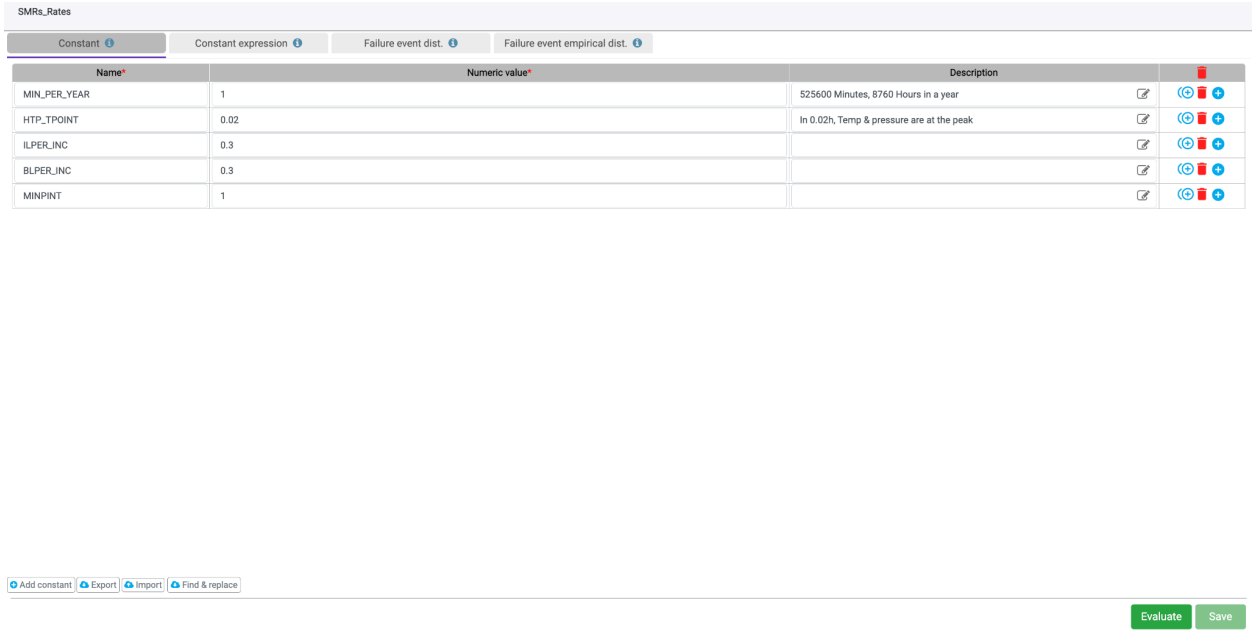
Export

Click the icon to export a parameter set in .ps format.

Update

Click a parameter set to update its details

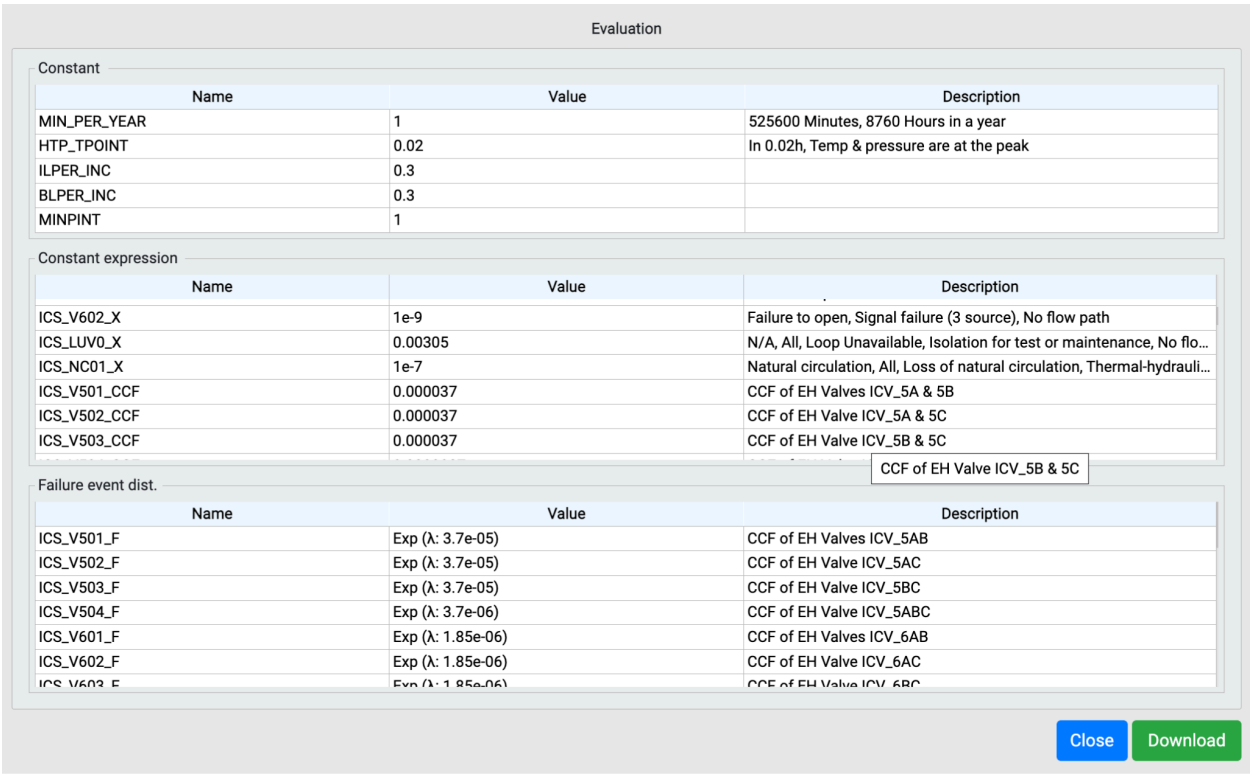
Constants



Constants can only be numeric e.g. 4, 2.3, 4e-6 etc. Their value can be changed at the time of analysis. For example, graphs can be plotted for matric results against ranges of values of constants.

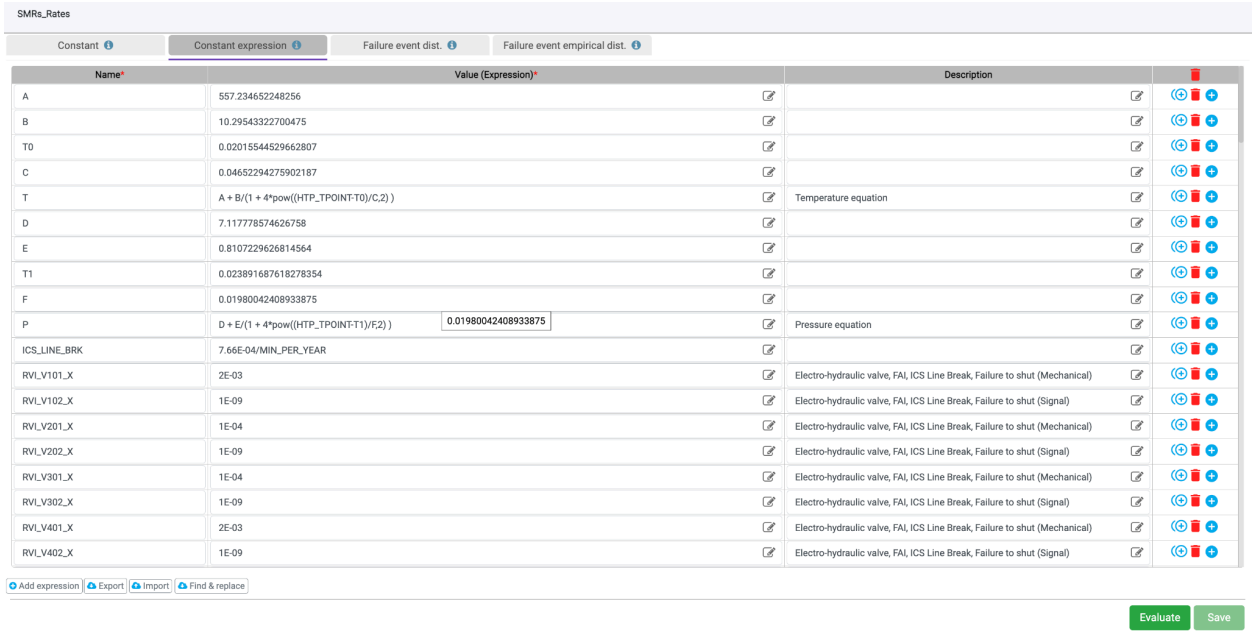
- Click the “Add constant” button to enter a new row in the table.
- Click the “Export” button to export constants in a CSV format.
- Click the “Import” button to import constants from a CSV file.
- Click the “Find & replace” button to search sub-strings among Constants and replace them with a new string.

- Click the  icon in the last column of any row to add a new row above it.
- Click the  icon to duplicate the row.
- Click the “Evaluate” button to evaluate expressions in all tabs. The result will be displayed in a popup.



- Click the “Save” button to save the data. This action will save data in all the tabs.

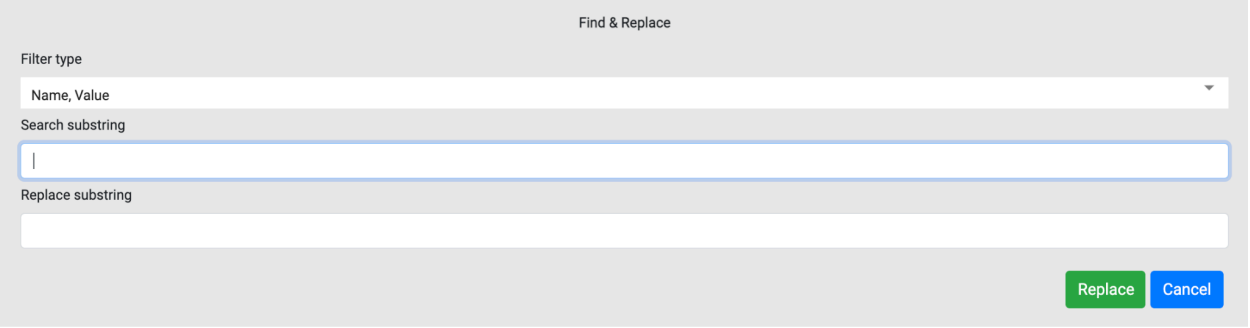
Constant Expressions



These are non-negative, real-value expressions, that can use constants (defined in the Constants tab) e.g. x + 2 where x is a constant. The grammar of the expression is given here.

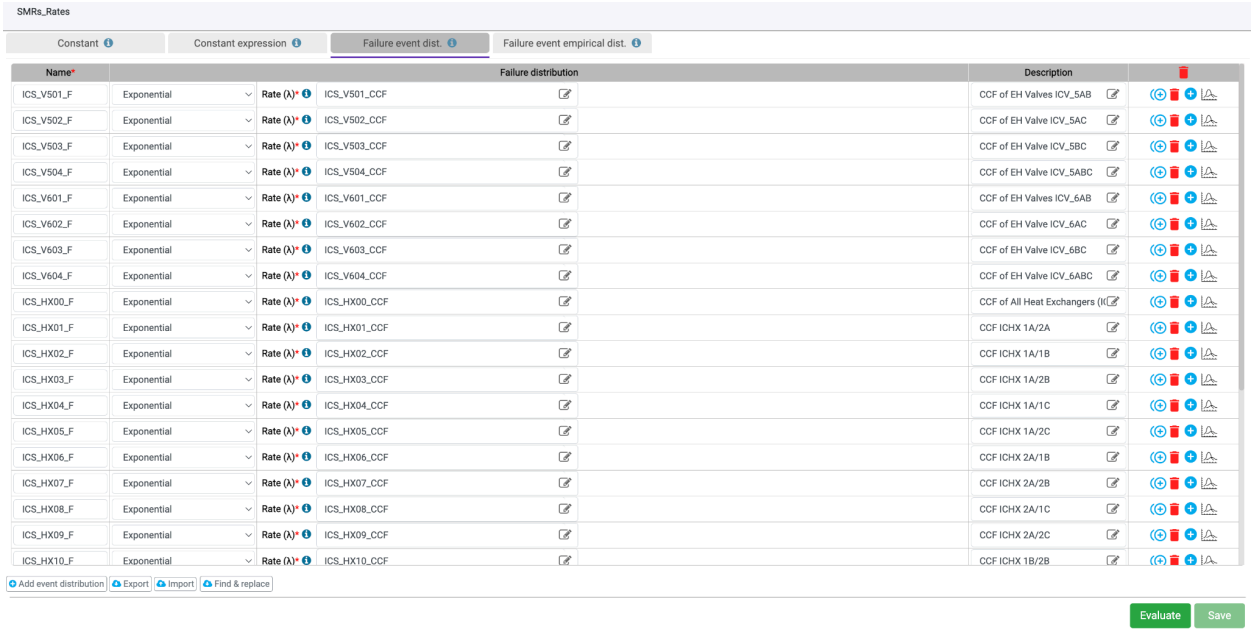
- Click “Add expression” to enter a new row in the table.
- Click “Export” to export expressions in a CSV format.

- Click the “Import” button to import expressions from a CSV file.
- Click the “Find & replace” button to search sub-strings among Constant Expressions and replace them with a new string.



- Click the icon to duplicate the row.
- Click the icon in the last column of any row to add a new row above it.
- Click the “Save” button to save the data. This action will save data in all the tabs.

Failure event distributions

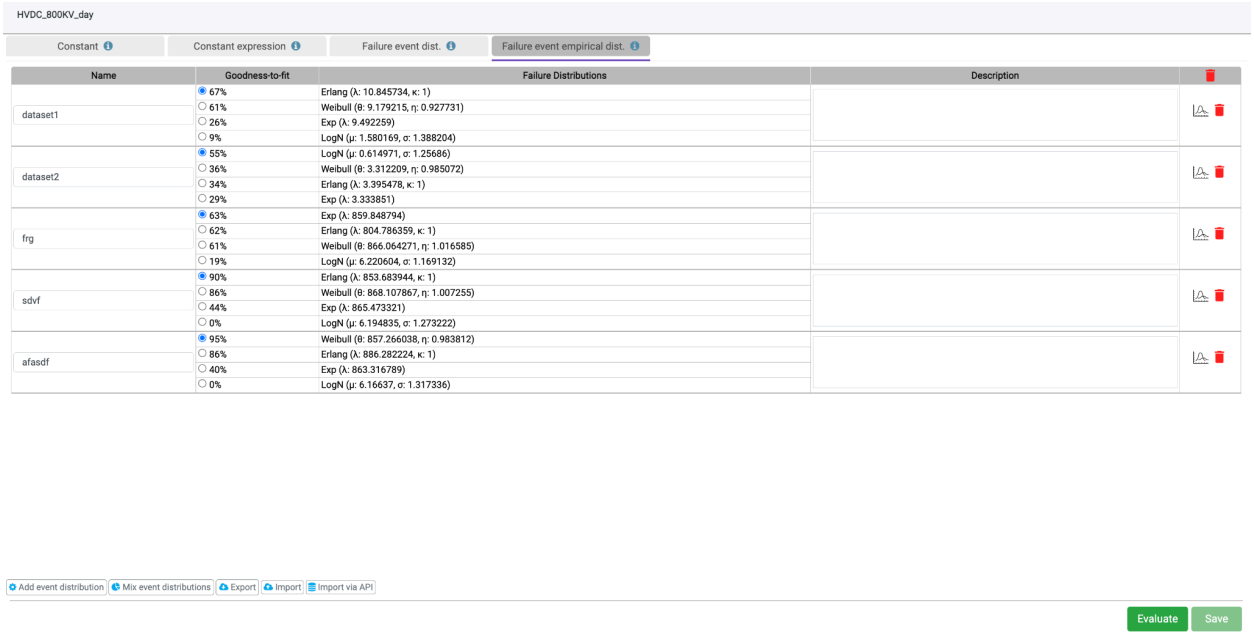


Failure event distributions can be exponential, erlang, Weibull, log-normal, and constant probability.

- Click the “Add event distribution” button to enter a new row in the table.
- Click the “Export” button to export failure distributions in a CSV format.
- Click the “Import” button to import failure distributions from a CSV file.
- Click the “Find & replace” button to search sub-strings among Constant Expressions and replace them with a new string.

- Click the icon to duplicate the row.
- Click the icon in the last column of any row to add a new row above it.
- Click the icon to view the graph of the corresponding distribution.
- Click the “Save” button to save the data. This action will save data in all the tabs.

Empirical failure distributions

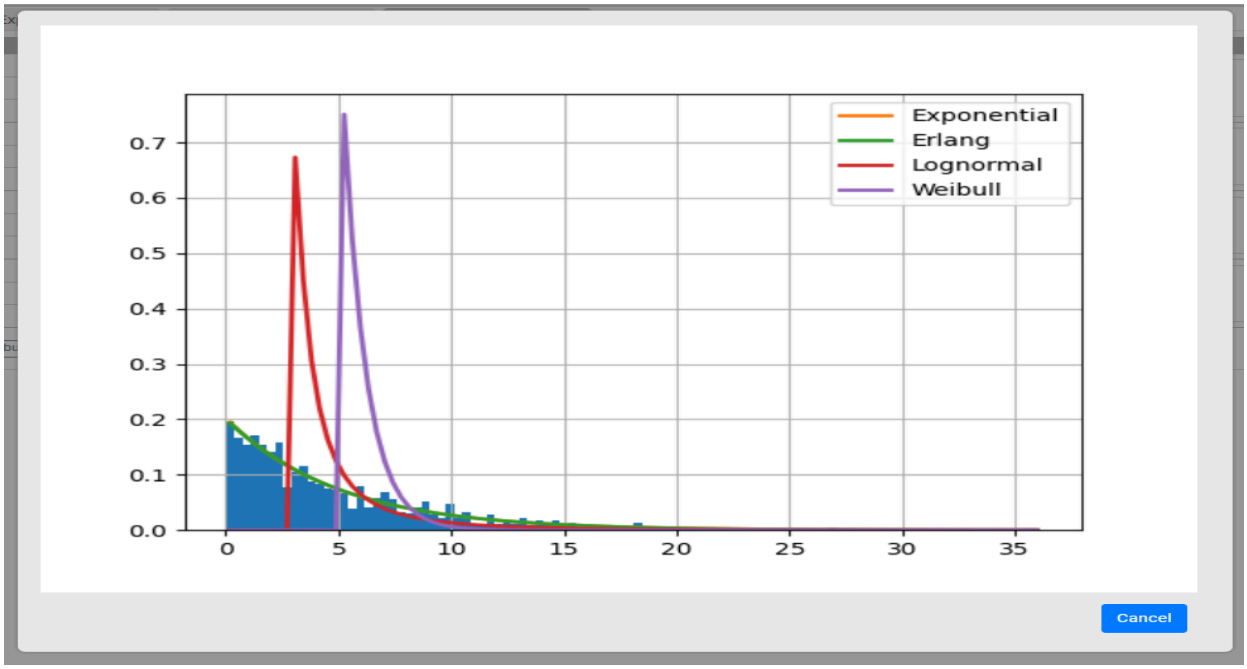


Empirical distributions are calculated from a data set using statistical methods. For that, historical failure data of a component is used to estimate the tentative failure probability distributions, which might have generated it, sorted according to their goodness-to-fit (GTF) values -- GTF value indicates the chance the data was generated by the corresponding distribution.

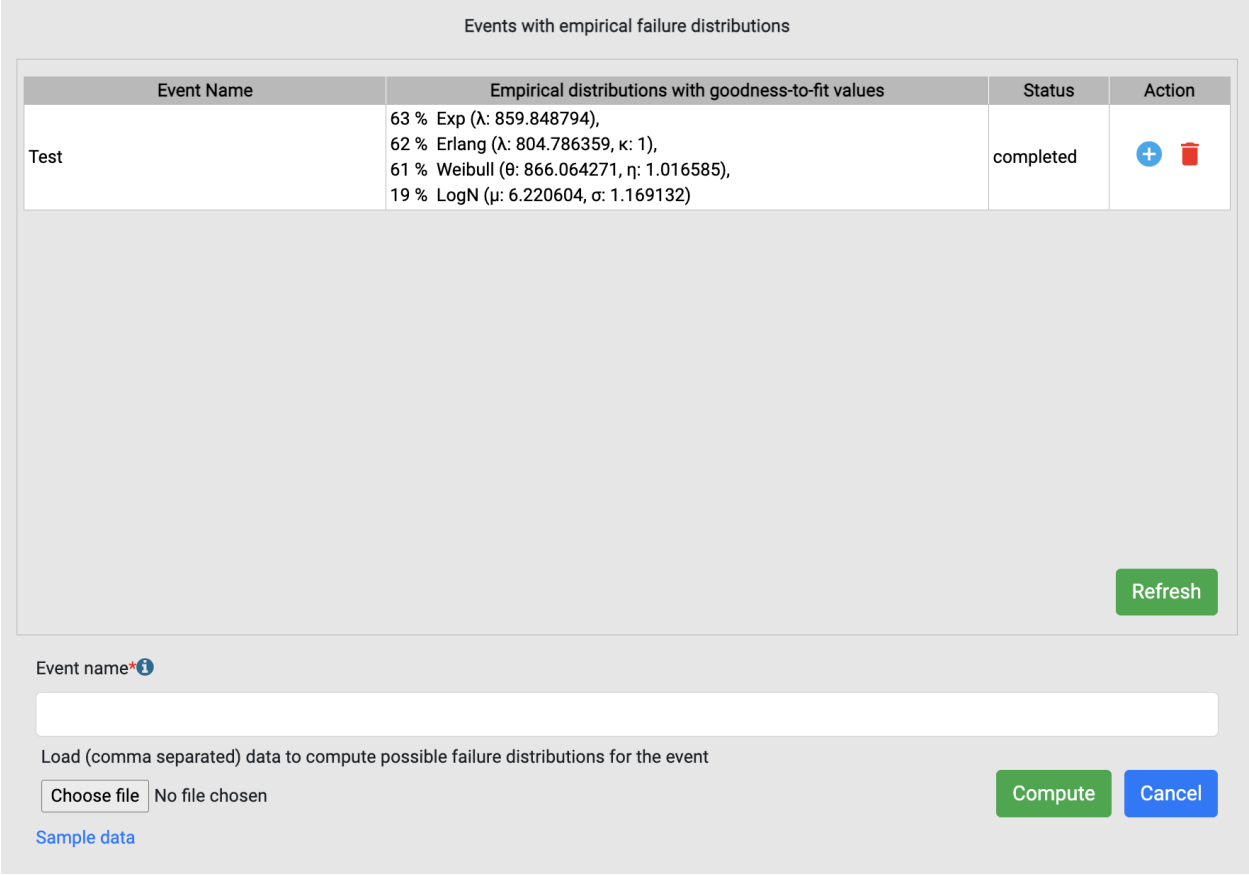
You can see how these distributions fit the data by clicking on the graph icon in the last column of their corresponding rows. Using Empirical distribution:

Each data may fit on multiple distributions, which are sorted according to their goodness-to-fit values, therefore we provide a radio button to select any distribution that we want to use.

Moreover, you can import and export empirical distributions (CSV format) and use them in other projects.

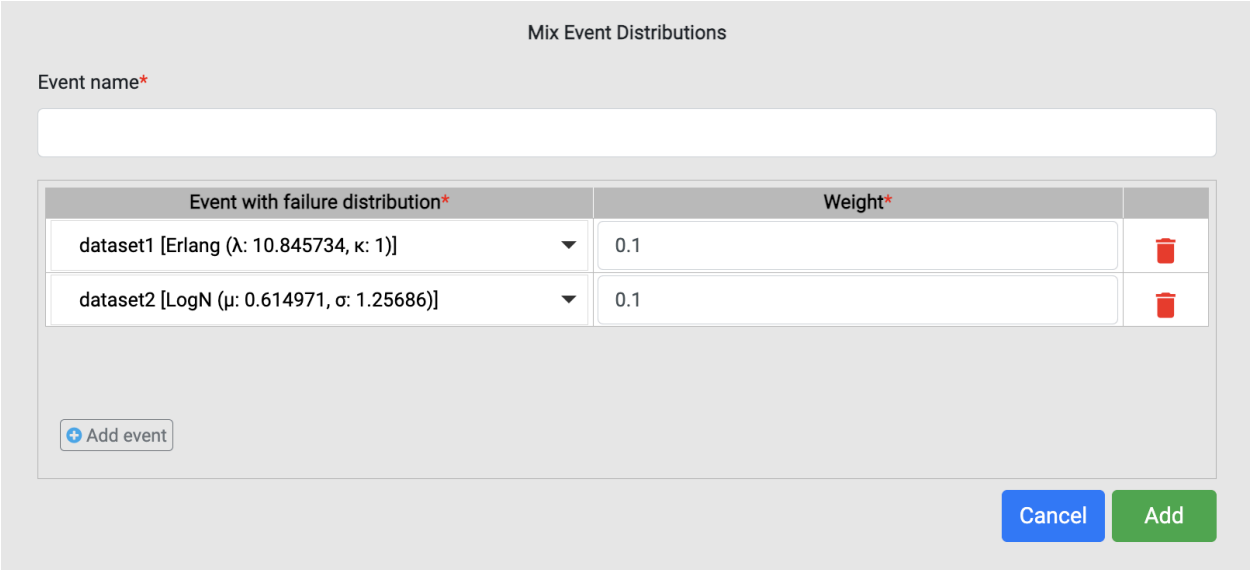


- **Generating empirical distribution.** An empirical distribution can be added by clicking the “Add event distribution” button. It will display all empirical distributions that have been generated previously and stored on the server side. One can compute a new distribution by specifying the file that contains data on which the distribution is to be approximated.

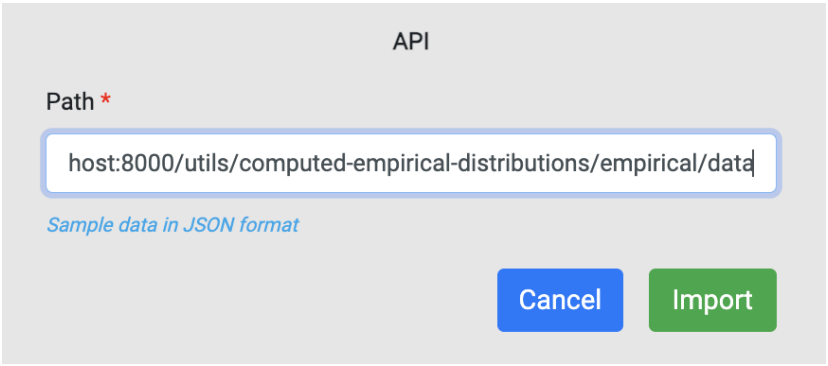


- In the “Events with empirical failure distributions” table, Click the icon in a row to add the distribution in the “Empirical Distributions” table in the main tab.
- To generate a new distribution from data, enter the name of the distribution in the “Distribution Name” field, choose the data file from your drive, and click the “Compute” button. A new row will appear in the “Computed Distributions” table with a “running” status.
- Click the “Refresh” button to check whether the distributions with “running” status have been computed.

- **Generating mixture distribution:** a mixture distribution can also be generated by clicking the “Mix distributions” button. It will show a popup where distributions along with their weights can be added. For example, d3 = 0.3*d1 + 0.7*d2 is a mixture distribution, where d1 and d2 are existing (empirical) failure distributions in the “Failure distributions” and “Empirical failure distributions” tabs. Click the “Add row” button if you want to add more distributions to the mixture.

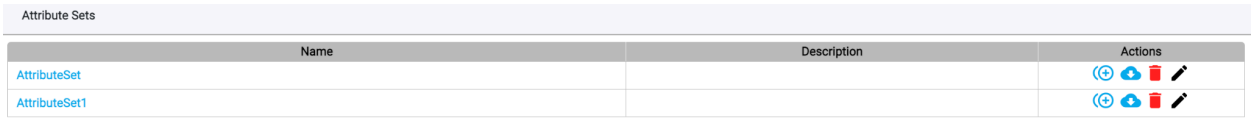


- Click the “Export” button to export empirical distributions in a JSON format (that includes the image data as well).
- Click the “Import” button to import empirical distributions in a JSON format (that includes the image data as well).
- Click the “Import via API” button to get empirical distributions that have been generated externally at a given API. The same data format can be accessed by clicking “Sample data in JSON format” on the popup.

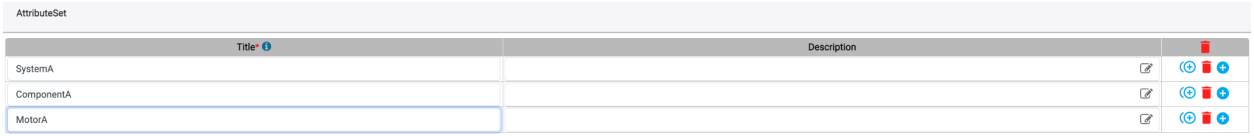


Attribute Sets

It contains sets of attributes that can be attached to fault tree elements. Click the “Attribute Sets” link in the left panel to open a page with all attribute sets thave have been defined.



- Click the icon to export an attribute set.
- Click the “Import attribute set” button to import an attribute set.
- Click the icon to duplicate an attribute set.
- Click the “Add attribute set” button to create a new attribute set.



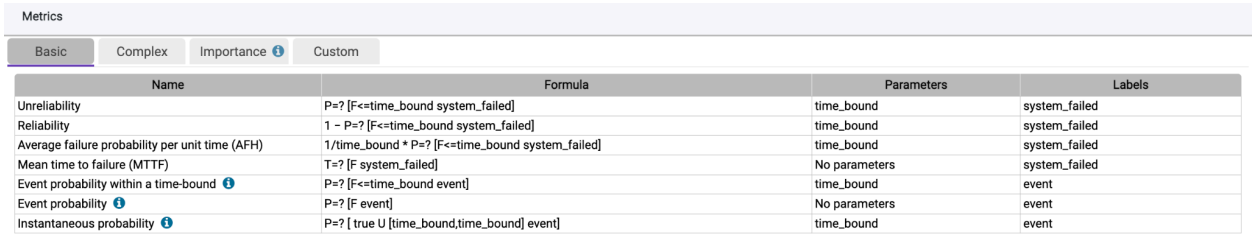
Click “Add attribute” to create a new attribute row in the table. Attributes can be imported/exported and duplicated by clicking the “Import”, and “Export” buttons, and the icon respectively.

Metrics

Metrics help us formally specify properties of interest we want to verify in fault trees. There is a list of predefined metrics (classified into basic, complex, and Importance metrics). For advanced users, it is possible to define custom metrics using continuous stochastic logic (CSL).

In advance view “Metrics” link is visible in the left panel. Click on it, and a screen with four tabs: Basic, Complex, Importance, and Custom will be visible.

Basic

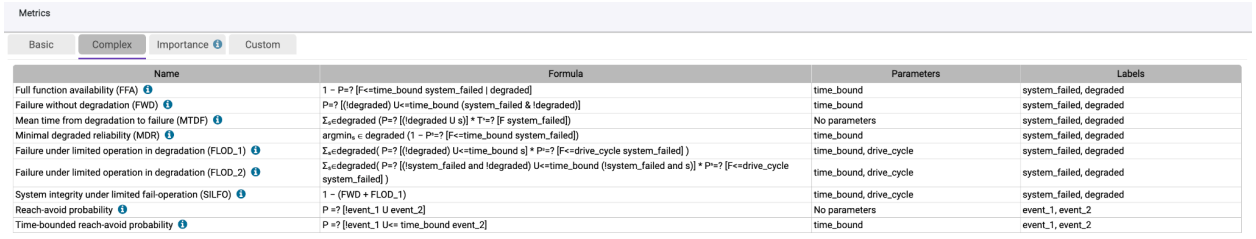


Name	Formula	Parameters	Labels
Unreliability	P=? [F<=time_bound system_failed]	time_bound	system_failed
Reliability	1 − P=? [F<=time_bound system_failed]	time_bound	system_failed
Average failure probability per unit time (AFH)	1/time_bound * P=? [F<=time_bound system_failed]	time_bound	system_failed
Mean time to failure (MTTF)	T=? [F system_failed]	No parameters	system_failed
Event probability within a time-bound	P=? [F<=time_bound event]	time_bound	event
Event probability	P=? [F event]	No parameters	event
Instantaneous probability	P=? [true U [time_bound,time_bound] event]	time_bound	event

It contains four important metrics that are verified in most of the reliability analysis cases:

- Reliability: Probability of failure in a given time bound.
- Unreliability: The complement of reliability (1- Reliability).
- Average failure probability per hour.
- Mean-time-to-failure. Expected time to system failure or scenario occurrence.
- Event probability within a time-bound: Probability that an event occurs within a given time.
- Event probability: Probability that an event occurs.
- Instantaneous probability: Probability that an event occurs at a given time.

Complex



Name	Formula	Parameters	Labels
Full function availability (FFA)	1 − P=? [F<=time_bound system_failed | degraded]	time_bound	system_failed, degraded
Failure without degradation (FWD)	P=? [(!degraded) U<=time_bound (system_failed & !degraded)]	time_bound	system_failed, degraded
Mean time from degradation to failure (MTDF)	$\Sigma_s$edegraded (P=? [(degraded U s)] * T=? [F system_failed])	No parameters	system_failed, degraded
Minimal degraded reliability (MDR)	$\text{argmin}_s \in$ degraded (1 − P=? [F<=time_bound system_failed])	time_bound	system_failed, degraded
Failure under limited operation in degradation (FLOD_1)	$\Sigma_s$edegraded(P=? [(!degraded) U<=time_bound s] * P=? [F<=drive_cycle system_failed])	time_bound, drive_cycle	system_failed, degraded
Failure under limited operation in degradation (FLOD_2)	$\Sigma_s$edegraded(P=? [(!system_failed and !degraded) U<=time_bound (!system_failed and s)] * P=? [F<=drive_cycle system_failed])	time_bound, drive_cycle	system_failed, degraded
System integrity under limited fail-operation (SILFO)	1 − (FWD + FLOD_1)	time_bound, drive_cycle	system_failed, degraded
Reach-avoid probability	P =? [levent_1 U event_2]	No parameters	event_1, event_2
Time-bounded reach-avoid probability	P =? [levent_1 U<= time_bound event_2]	time_bound	event_1, event_2

Some of these metrics cannot be verified directly by the Storm model-checker. They need some additional computations for their verification.

- Full Function Availability (FFA) describes the time-bounded probability that the system provides full functionality, i.e., it has neither failed nor degraded. It is described as the complement of the time-bounded reachability of a failed or degraded state.
- Failure Without Degradation (FWD) describes the time-bounded probability that the system fails without being degraded first. It is the time-bounded reach-avoid probability of reaching a failed state without reaching a degraded state.

- Mean Time from Degradation to Failure (MTDF) describes the expected time from the moment of degradation to system failure. It is obtained by taking the expected time of failure for each degraded state and scaling it with the probability of reaching this state while not being degraded before.
- Minimal Degraded Reliability (MDR) describes the criticality of degraded states by giving the worst-case failure probability when using the system in a degraded state. For all degraded states, the time-bounded reachability of a TLE failure is computed. The MDR is the minimum over the complement of this result for all degraded states.
- Failure under Limited Operation in Degradation (FLOD_1) describes the probability of failure when imposing a time limit for using a degraded system. For all degraded states, the time-bounded reachability probability of a failed state is computed within the restricted time-bound given by a drive cycle. This value is scaled by the time-bounded reach-avoid probability of reaching a degraded state without degradation before.
- Failure under Limited Operation in Degradation (FLOD_2) describes the probability of failure when imposing a time limit for using a degraded system. For all degraded states, the time-bounded reachability probability of a failed state is computed within the restricted time-bound given by a drive cycle. This value is scaled by the time-bounded reach-avoid probability of reaching a degraded state without degradation or system failure before.
- System Integrity under Limited Fail-Operation (SILFO) considers the system-wide impact of limiting the degraded operation time. SILFO is split into two parts considering failures without degradation (FWD) and failures with degradation (FLOD_1).
- Reach-avoid probability: Probability to occur an event, say e1, without occurring another event, say e2, before.
- Time-bounded reach-avoid probability: Probability to occur an event, say e1, within a time_bound without occurring another event, say e2, before.

Importance

Metrics

Basic | Complex | Importance | Custom

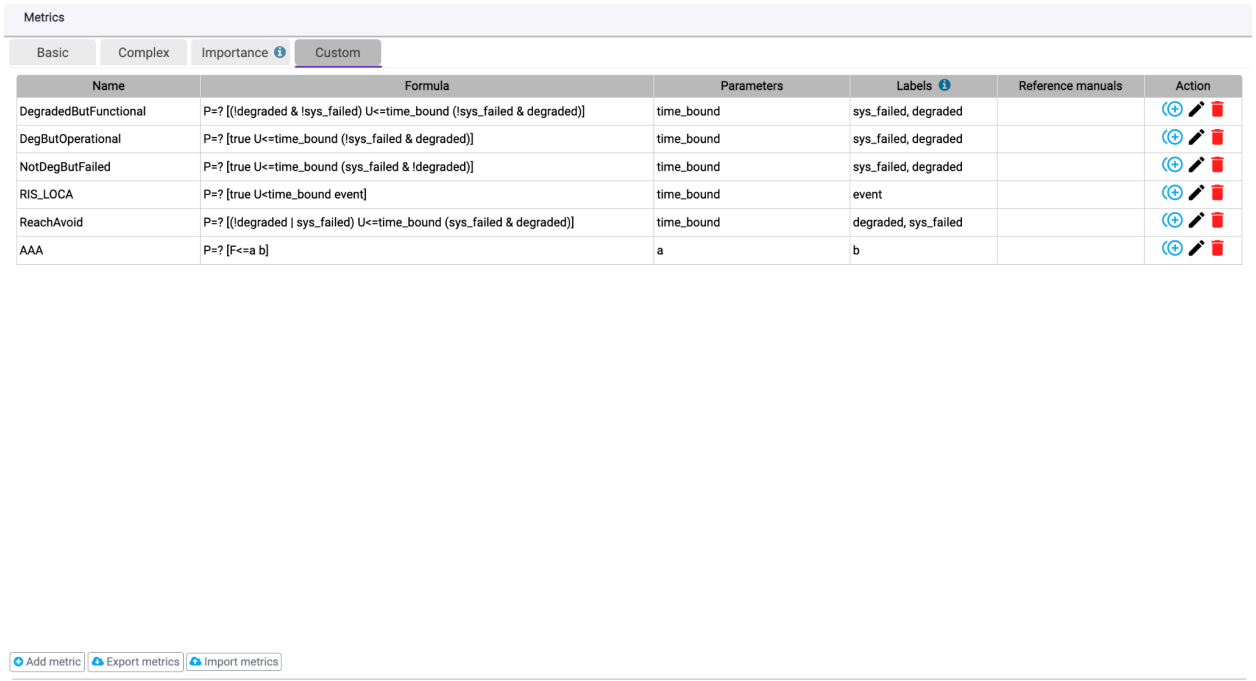
Name	Formula	Parameters	Labels
Birnbaum Index (BI)	∂ Unr(system_failed, time_bound) / ∂ Unr(component_failed, time_bound)	time_bound	system_failed, component_failed
Criticality Importance (CI)	BI * Unr(component_failed, time_bound) / Unr(system_failed, time_bound)	time_bound	system_failed, component_failed
Risk Achievement Worth (RAW)	Unr(system_failed [ρ(component_failed) = 1], time_bound) / Unr(system_failed, time_bound)	time_bound	system_failed, component_failed
Risk Reduction Worth (RRW)	Unr(system_failed, time_bound)/ Unr(system_failed [ρ(component_failed) = 0], time_bound)	time_bound	system_failed, component_failed
Diagnostics Importance Factor (DIF)	Unr(system_failed & component_failed, time_bound)/ Unr(system_failed, time_bound)	time_bound	system_failed, component_failed
BAGT+	| MTTF(system_failed) - MTTF (system_failed [ρ(component_failed) = 1)|	No parameters	system_failed, component_failed
BAGT-	| MTTF(system_failed) - MTTF (system_failed [ρ(component_failed) = 0)|	No parameters	system_failed, component_failed

These metrics cannot be verified directly by the Storm model-checker. They need some additional computations for verification. Note that different importance measures often give different results. In general, this does not mean that one of them gives a wrong result; rather, they measure different aspects of importance. Therefore, it is recommended to consider multiple importance measures and combine their results to have a stronger understanding of the importance of each component.

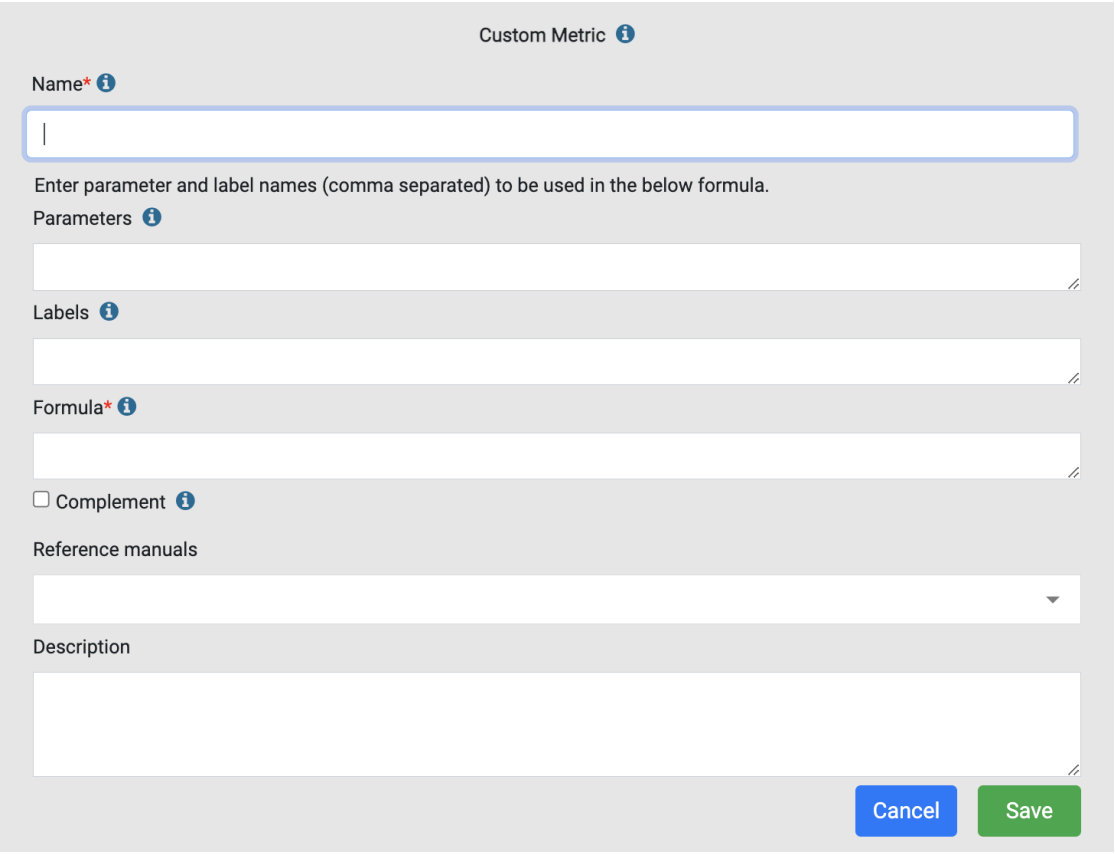
- Birnbaum Index (BI): How much the unreliability of the system depends on the unreliability of a specific component.

- Criticality Importance (CI): How much the unreliability of the system depends on the unreliability of a specific component scaled by the ratio of component and system unreliability.
- Risk Achievement Worth (RAW): The impact of a component's total degradation on system unreliability.
- Risk Reduction Worth (RRW): The impact of making the component fully reliable on the system's unreliability.
- Diagnostics Importance Factor (DIF): How often a component fails in states where the system has failed.
- BAGT+: Change in MTTF if the component fully degrades.
- BAGT-: Change in MTTF if the component is fully reliable.

Custom



One can create custom metrics on the “Custom” tab. It allows specifying metrics using continuous stochastic logic (CSL).



- Click the “Add metric” button to add a new metric.
    - Parameters and labels used inside metrics formulae must have unique names among themselves, starting with a letter or underscore (_) followed by underscores, letters, and/or numbers. They must not be from the list of keywords - - true, false, Pin, Pmax, Smin, Smax, Tmin, Tmax, LAmin, LRAmax, P, R, T, S, LRA, min, max, G, U, F, W, C, I, failed.
    - The formula can be defined using probabilistic computation tree logic (PCTL)/continuous stochastic logic (CSL). For example, P = ? [true U <=10000 (failed & ! mode1)], where failed and mode1 represent quantifiable states. The grammar of expressions is given here.
    - The “Complement” checkbox can be selected to calculate the complement of a property mentioned in the formula field. It only makes sense for properties that calculate probabilities of events.
    - Reference manuals can be attached with a metric.
    - Note that metric parameters which are entered on the above screen are exclusively dedicated to metrics. Their values are not taken from the parameter set that is attached to a failure model. However, their values can be changed at the time of analysis, and ranges for their values can be provided to draw plots.
    - Click the “Export metrics”/”Import metrics” to export/import metrics in the .metrics format (a format supported by our SAFEST tool).

Computing

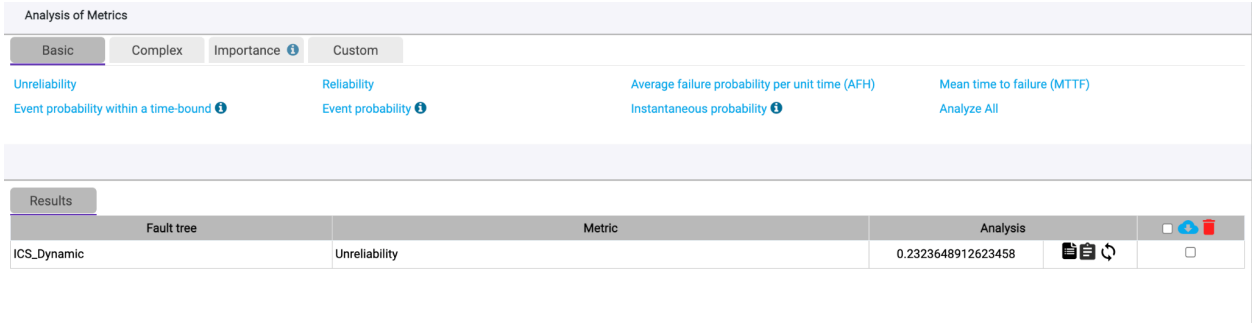
Failure models can be analyzed in different ways:

- Analysis – the exact results of different metrics can be computed.
- Bounded Analysis – the exact values of metrics can be bounded from above and below graphically.
- Graphs – the exact results of different metrics can be graphed against e.g. time.
- Interactive Simulation – failure propagation in fault trees can be shown interactively on multiple pages at the same time.
- Minimal Cut Sets – for static fault trees MCS can be computed and displayed graphically.

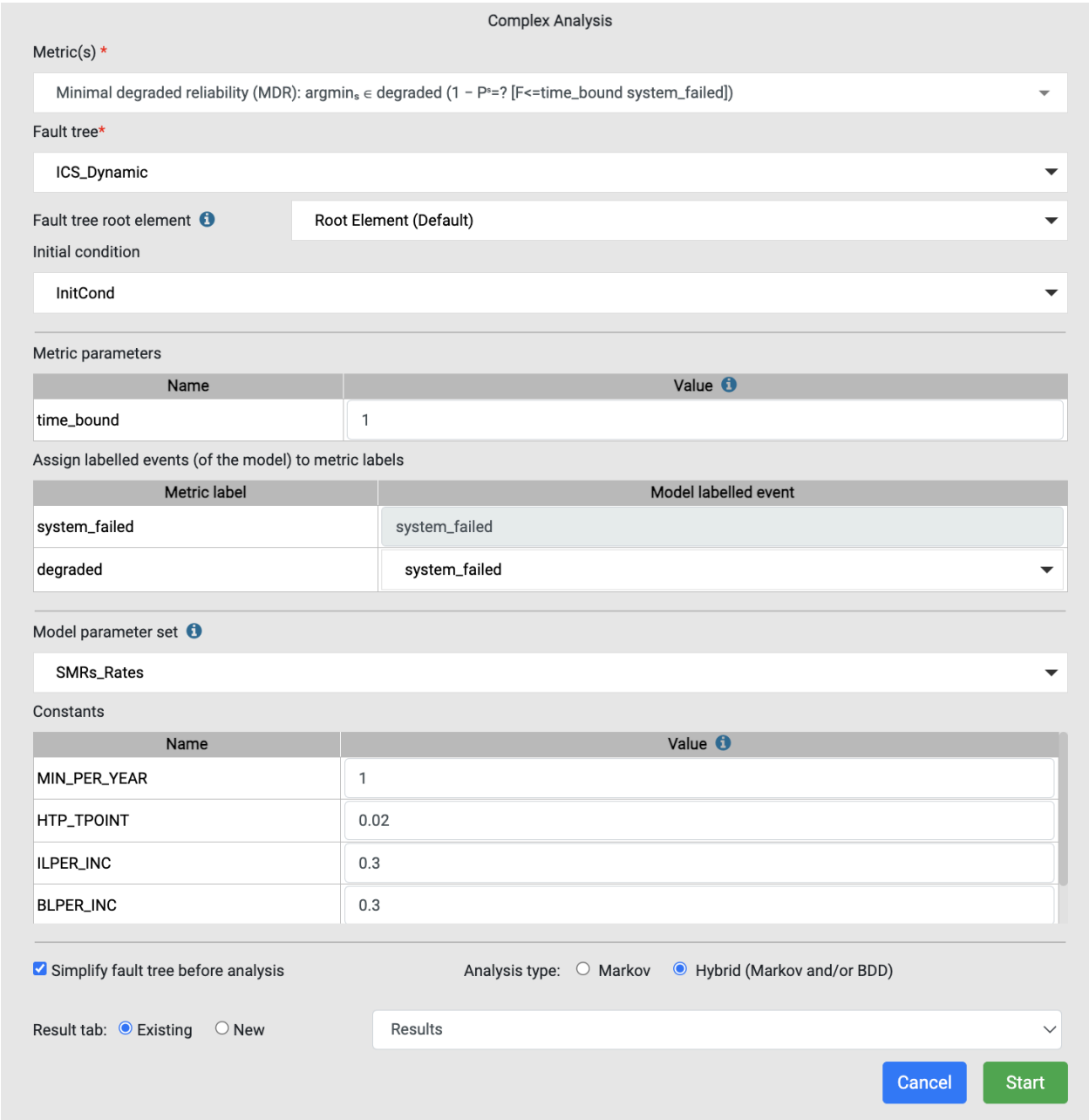
(Exact) Analysis

Complex systems usually have dynamic behavior because of e.g. spare components, failure sequence among components, functional dependencies, etc. The analysis of such systems is quite complex which is usually based on simulation or generalization techniques. Unlike others, we implement formal verification techniques e.g. probabilistic model-checking, and thus provide exact results on measures of interest.

Click on the “Analysis” link under “Computing” in the left panel. The following window will appear with four tabs for different classes of metrics.

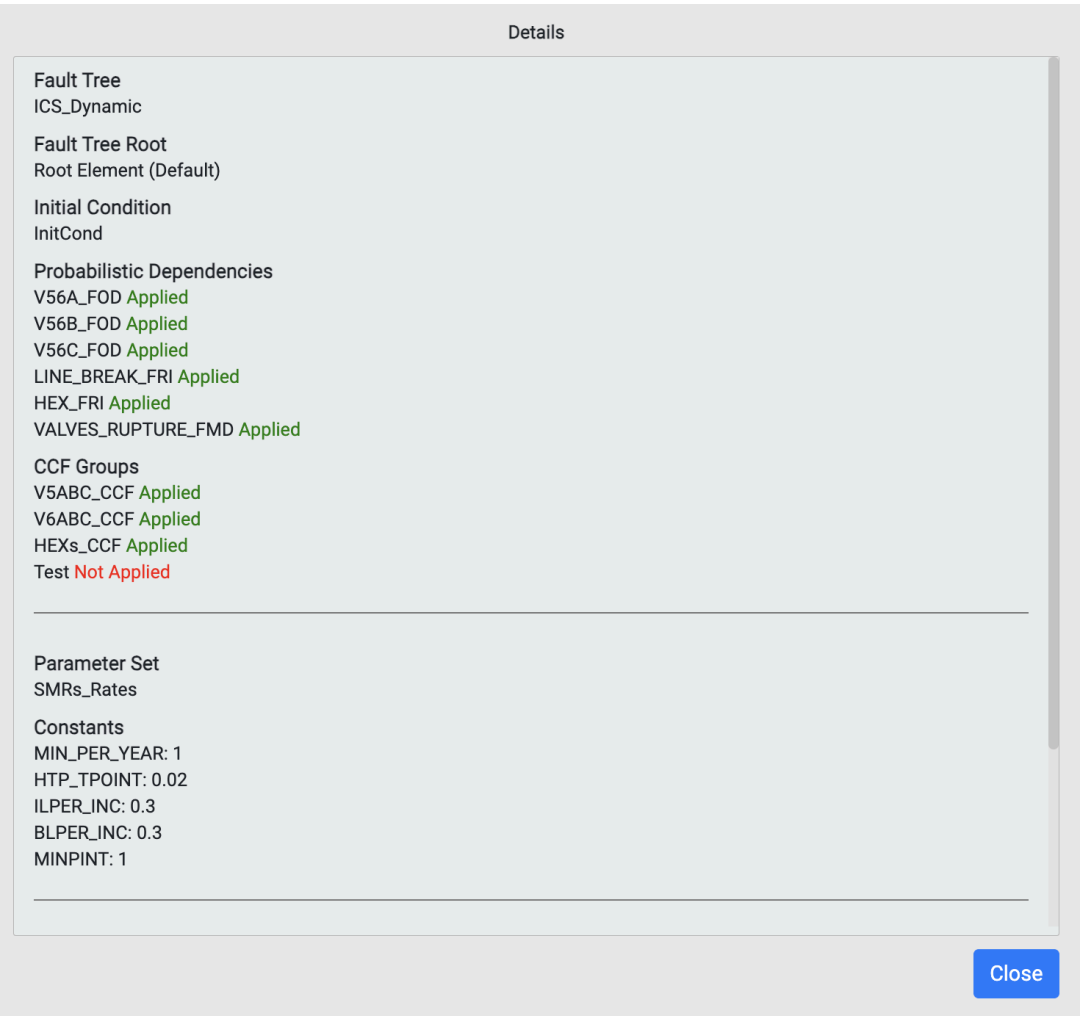


- One can verify a metric on each tab, the mechanism is more or less the same. For example, click on the “Minimal degraded reliability (MDR)” link on the complex tab. The following window will appear:

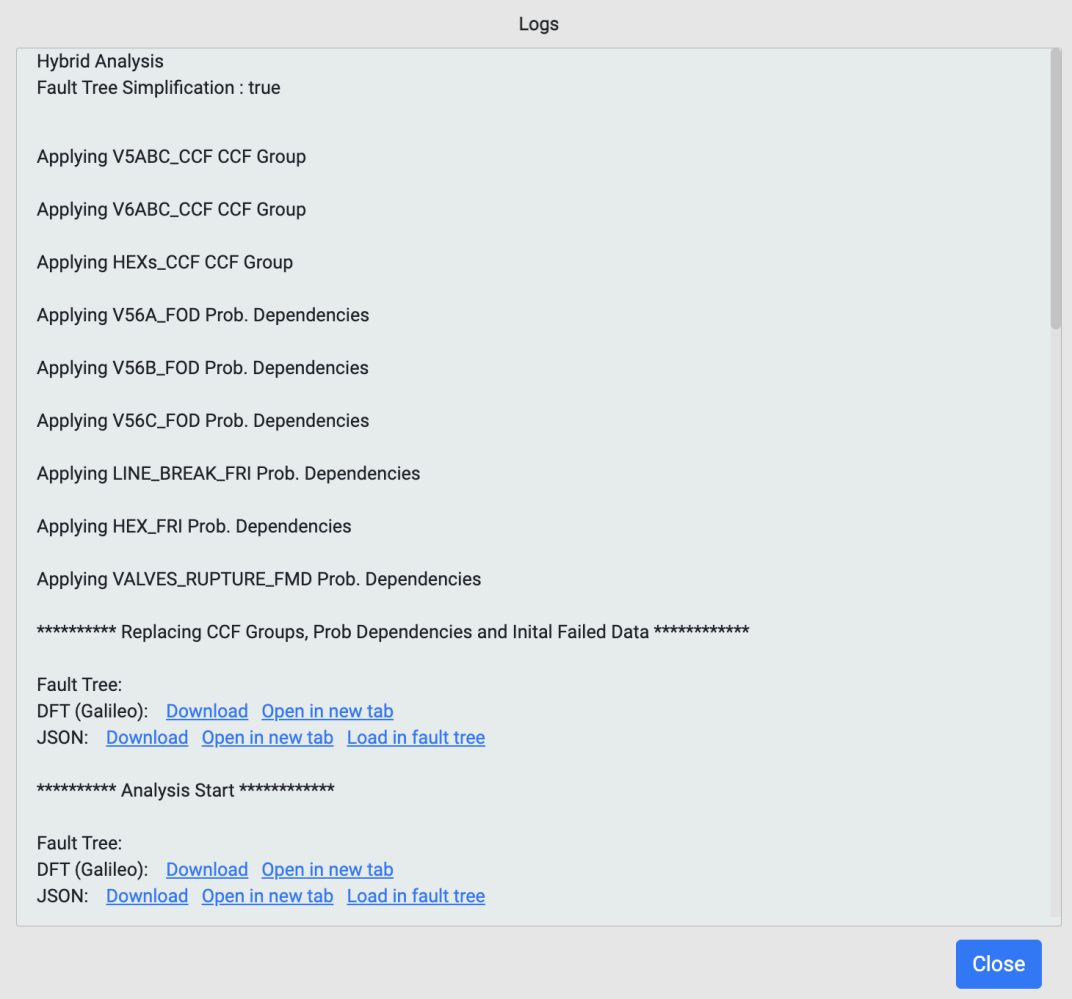


- “Fault tree” dropdown: a fault tree that is selected as a default model in the “Fault trees” window is automatically selected.
- “Fault tree root element”: the root element of the above-selected fault tree is selected by default. One can change the root element by selecting any other element in the dropdown. Note that the dropdown contains only those elements of the fault tree whose “Generate label for the failure event” checkbox is selected in the fault tree.
- Select the initial condition that you want to apply to the fault tree.
- “Metric parameters”: Note that metric parameters cannot take values defined in the parameter set attached to the above-selected fault tree. Each metric parameter has to be assigned a value. For the “time-bound” metric parameter, a

default value is assigned to it that is associated with the above-selected fault tree.
    - “Assign labelled events (of the model) to metric labels”: Assign failure events of the fault tree to metric labels.
    - A parameter set that is attached to the selected fault tree (above) is automatically selected. It can be changed at this point to generate another variant of the fault tree.
    - Optionally, change the values of “Constants” defined in the selected parameter set. Note that values of other elements of the parameter set (real-value expressions, (empirical) failure distributions) cannot be changed at the time of analysis.
    - The analysis method is selected automatically based on the selected metric. However, for some metrics both Markov and Hybrid (BDD and/or Markov) analysis are possible. One can decide the analysis type by selecting the corresponding radio button.
    - “Simplify fault tree before analysis”: select this checkbox if the fault tree has to be simplified by applying simplification rules before analysis.
    - Finally, select a tab on which the result of the analysis has to be displayed. It can be an existing tab or a new tab.
- Click the  icon to view configuration of the analysis.



- Click the icon to view the analysis log.



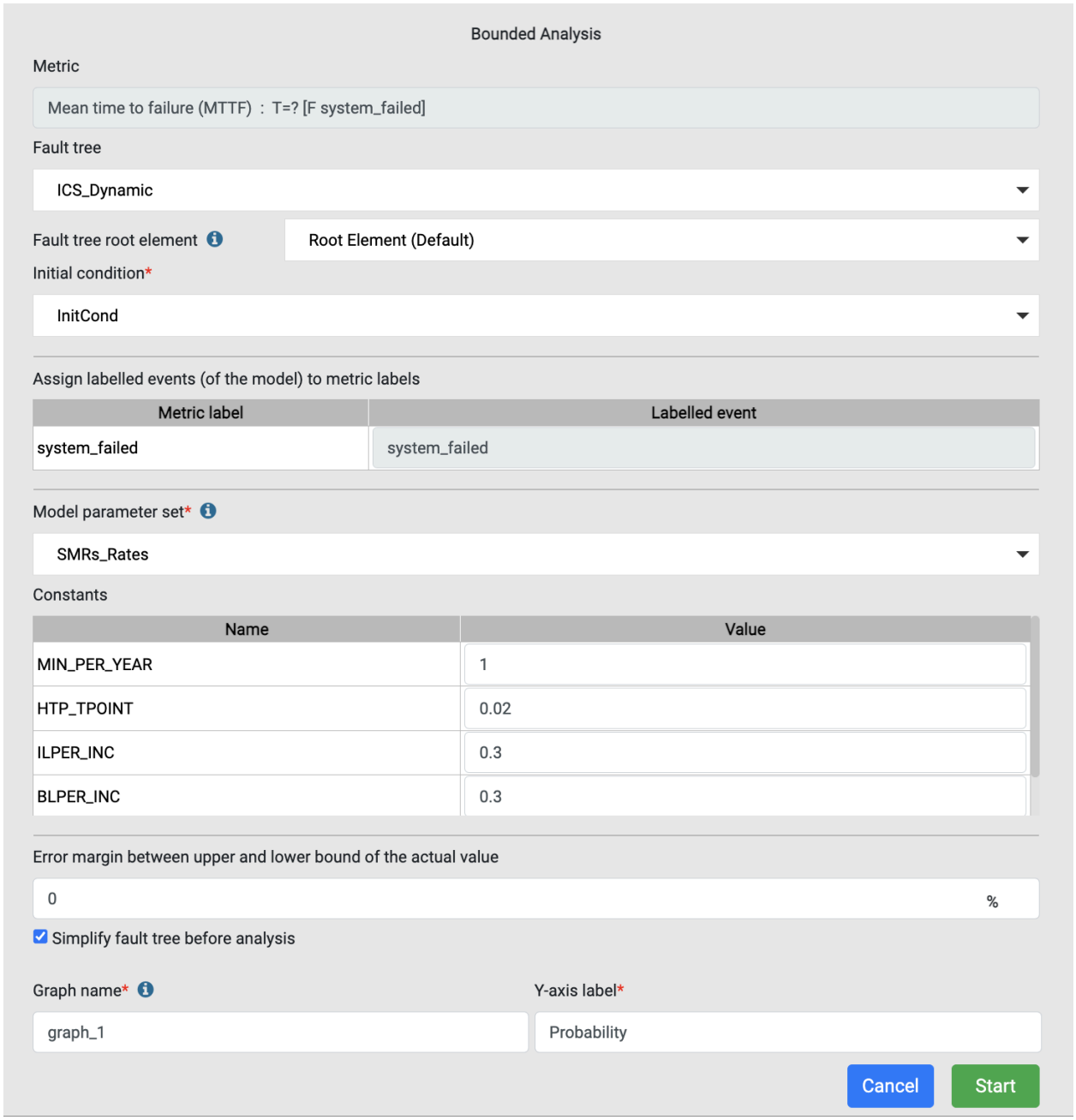
  - Click the “Download” link to download the generated artifact (DFT/DRN).
  - Click the “Open in new tab” link to open the generated artifact in a new browser tab.
  - Click the “Load in fault tree” link to load the generated fault tree in the current project.
- Click the  icon to rerun the analysis with selected configurations.
- Click  icon to download the results in the selected rows in a CSV format.

Bounded analysis

To compute exact results for measures using Markov analysis, first of all, the full state space is constructed and then analyzed. However, many states in the state space only marginally contribute to the result. If one is interested in an approximation of the MTTF (or the reliability), these states are of minor interest. We implemented the algorithms, proposed by Dr. Matthias Volk et. al., that generate state space on-the-fly, and then compute an upper and a lower bound to the exact results on a partially unfolded system, which might be much smaller as compared to

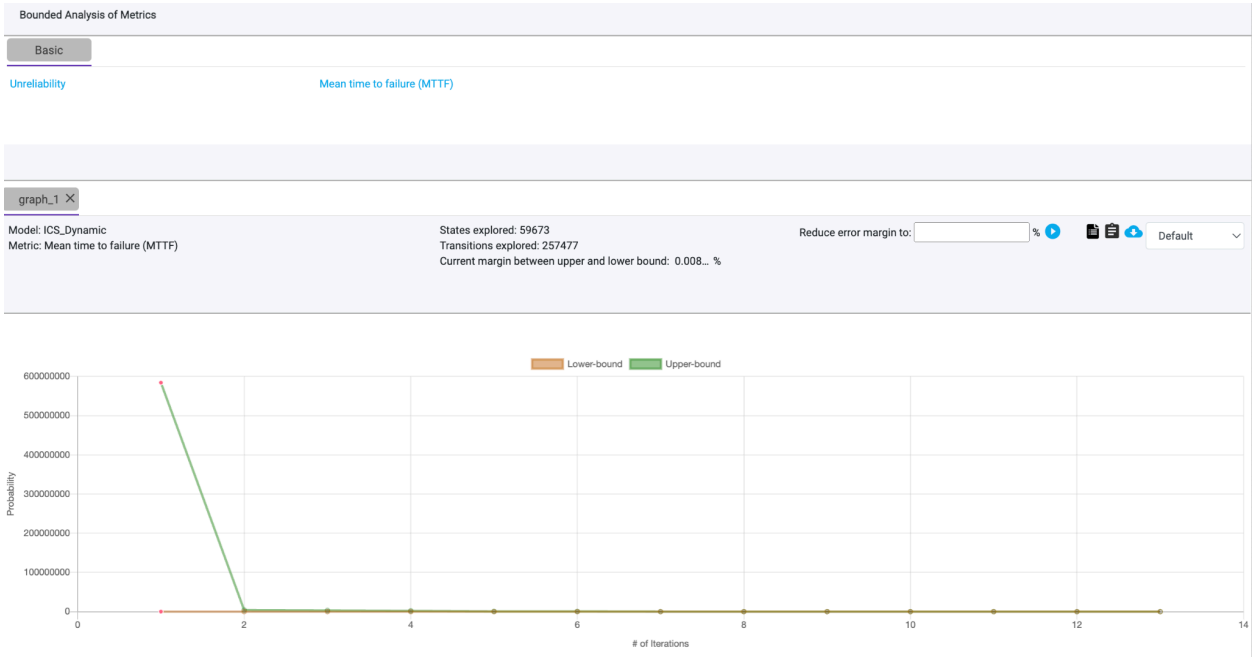
the fully unfolded system. The approximation is sound ensuring the exact result lies between these two bounds.

Click on the “Bounded Analysis” link under “Computing” in the left panel and then click e.g. “Mean-time-to-failure” link. The following window will appear:



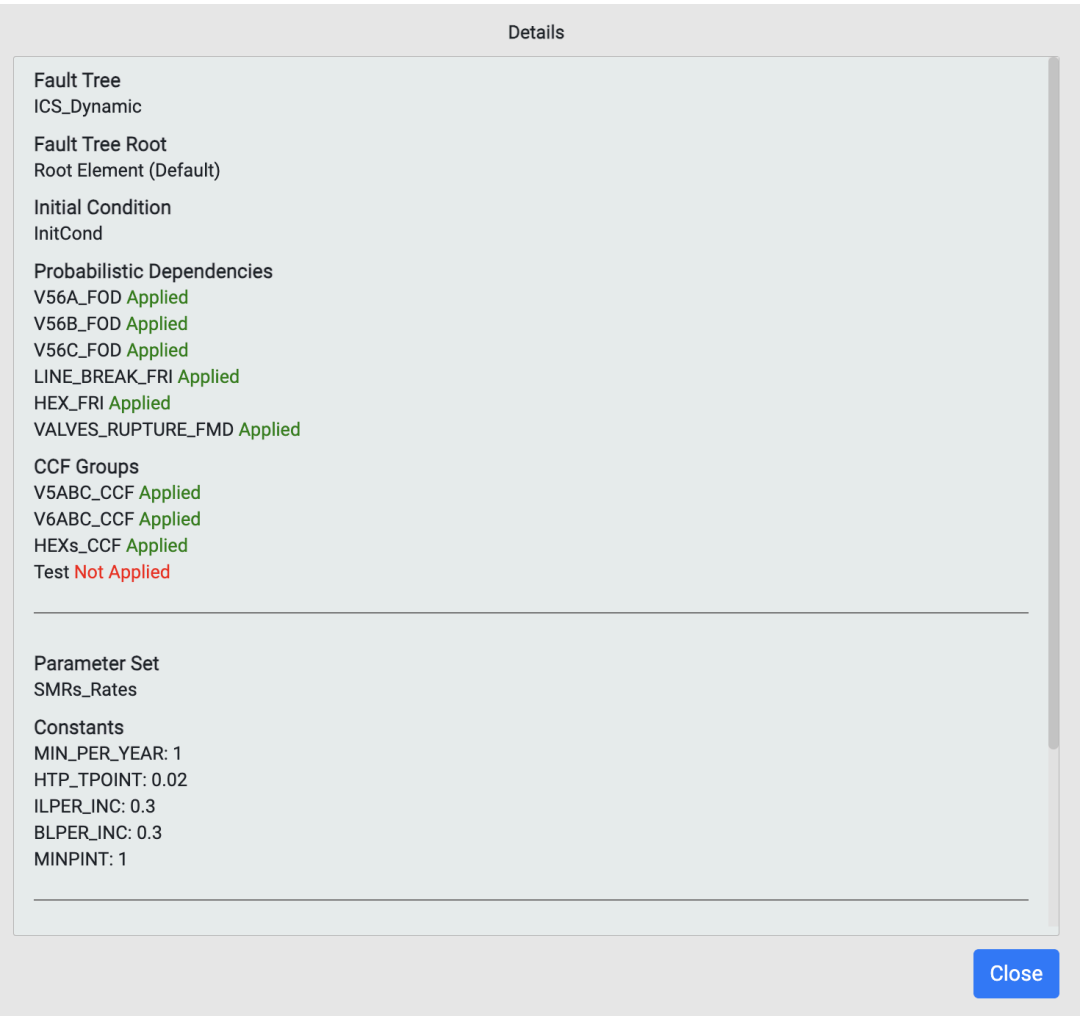
- All fields are filled up as described in the “Analysis” case with a few additions:
- “Error margin between upper and lower bound of the actual value”: A percentage error margin is entered in this field.

- Optionally enter “Graph name” and the label of its Y-axis. Note X-axis will always represent the number of iterations in this case.
- Bounded analysis is always done by the Markov technique.
- Click the “Start” button, the results will be displayed as:

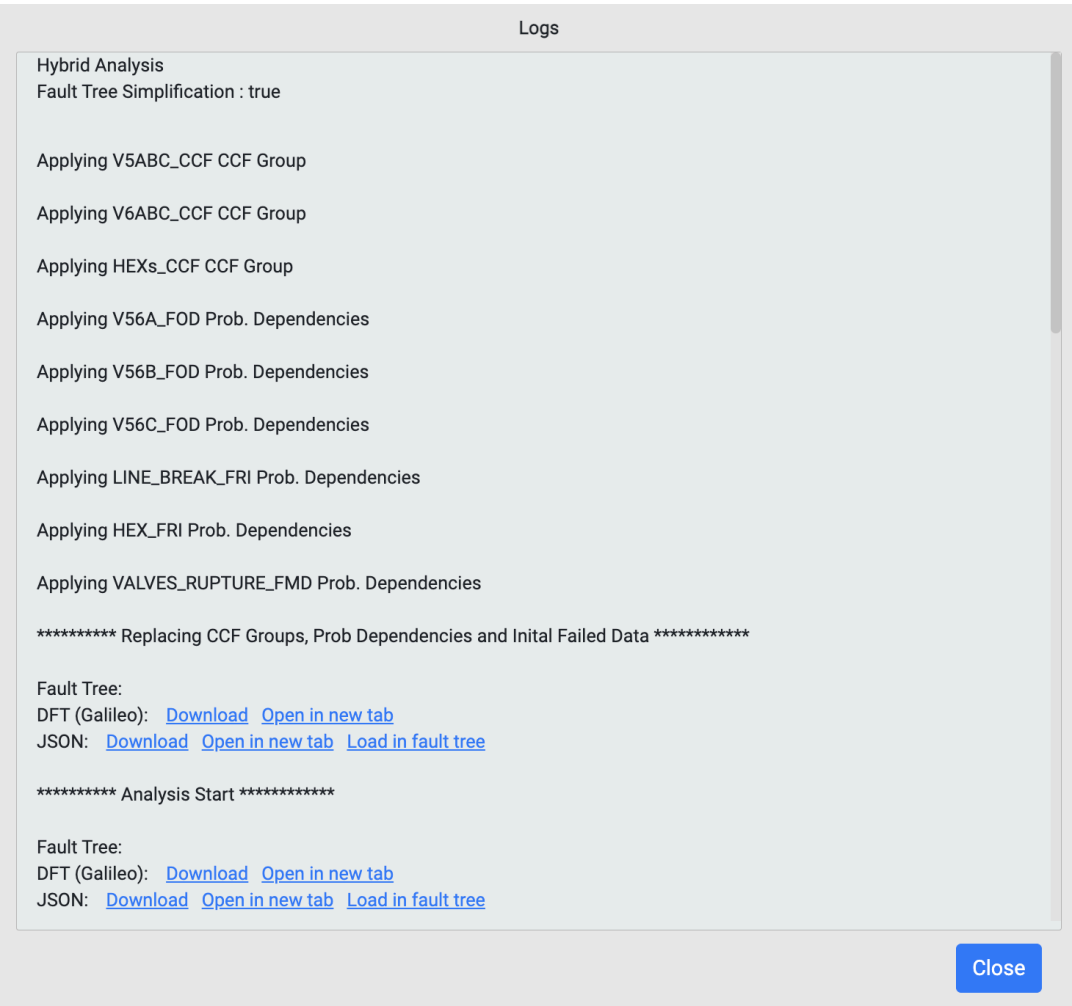


The upper line in the graph shows the upper bound whereas the lower line shows the lower bound on the actual value of the metric.

- In addition, we show the number of generated states and the transitions explored so far.
- In case you are interested in further reducing the error margin, insert a new value in the text field and click the play button  .
- One can apply a log function on the values of the Y-axis by selecting it on the right side of the graph.
- The graph values can be downloaded by clicking on the  icon.
- Click the  icon to view configuration of the analysis.



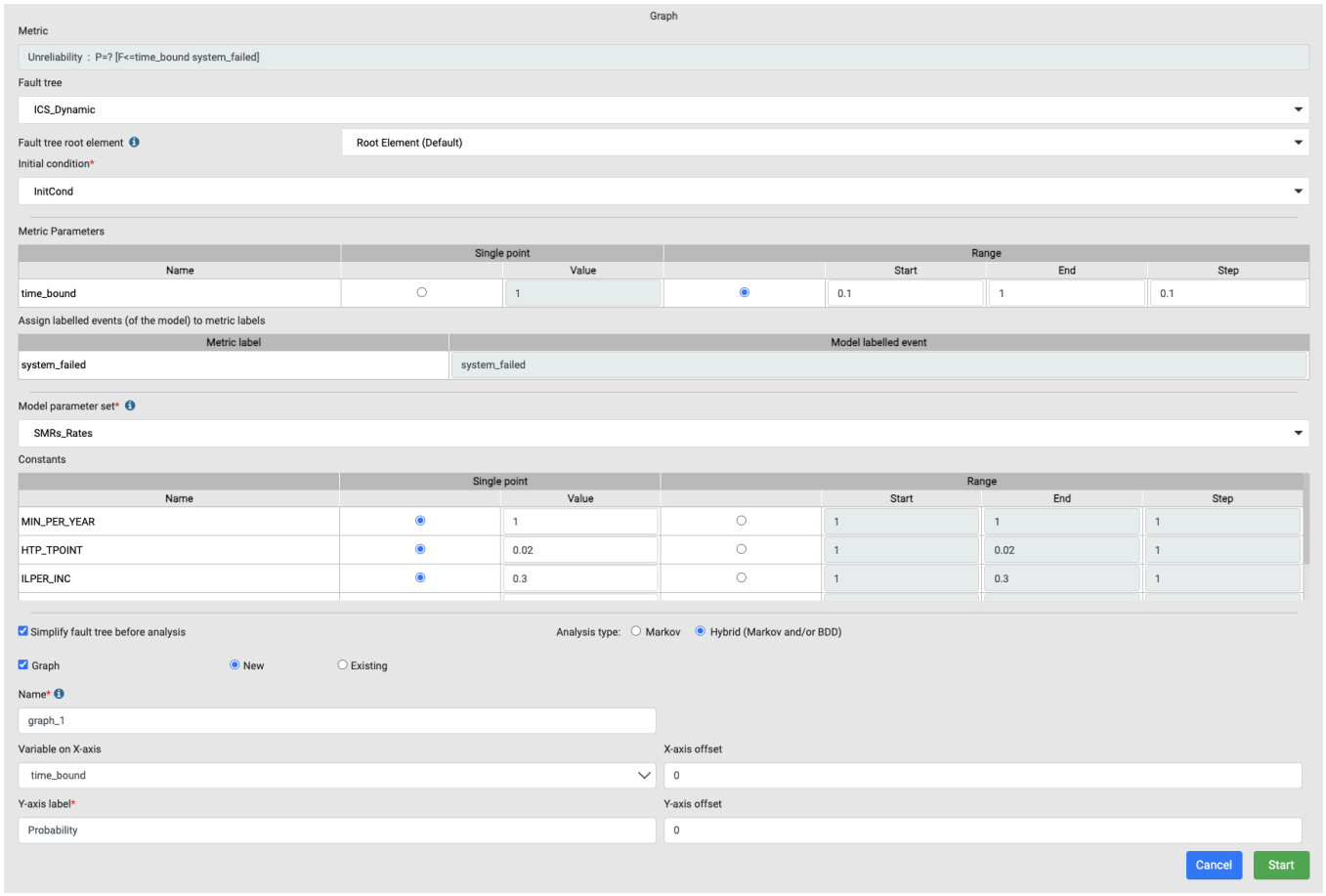
- Click the icon to view the analysis log.



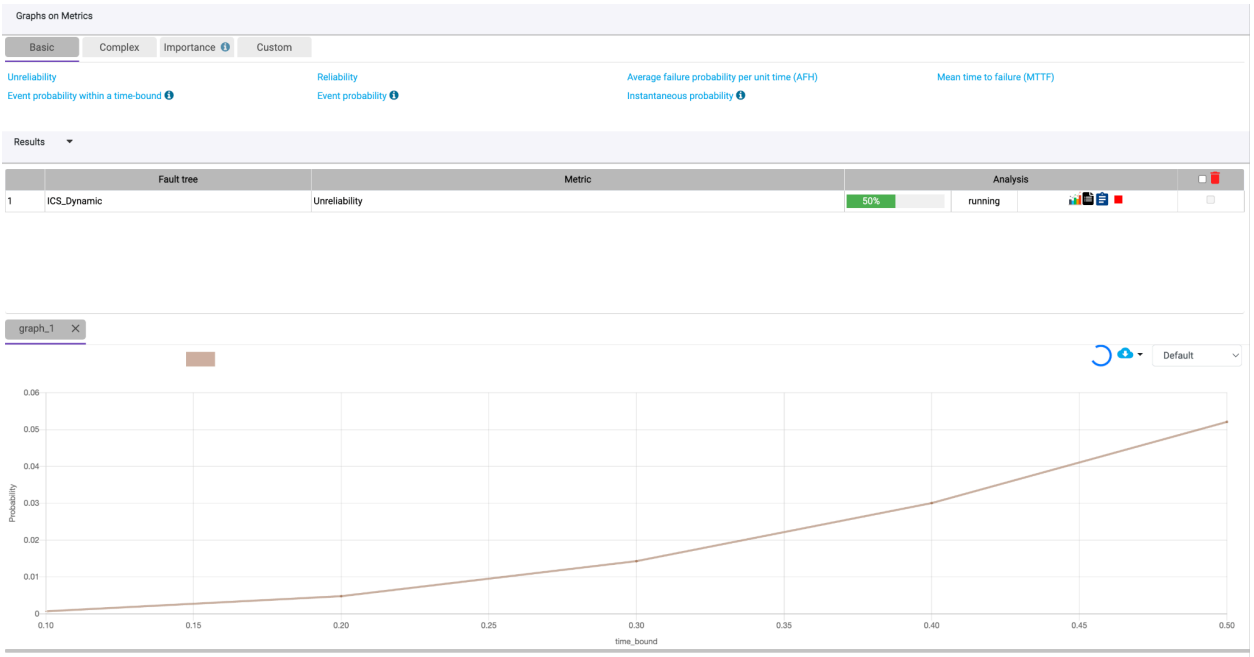
- Click the “Download” link to download the generated artifact (DFT/DRN).
- Click the “Open in new tab” link to open the generated artifact in a new browser tab.
- Click the “Load in fault tree” link to load the generated fault tree in the current project.

Graphs

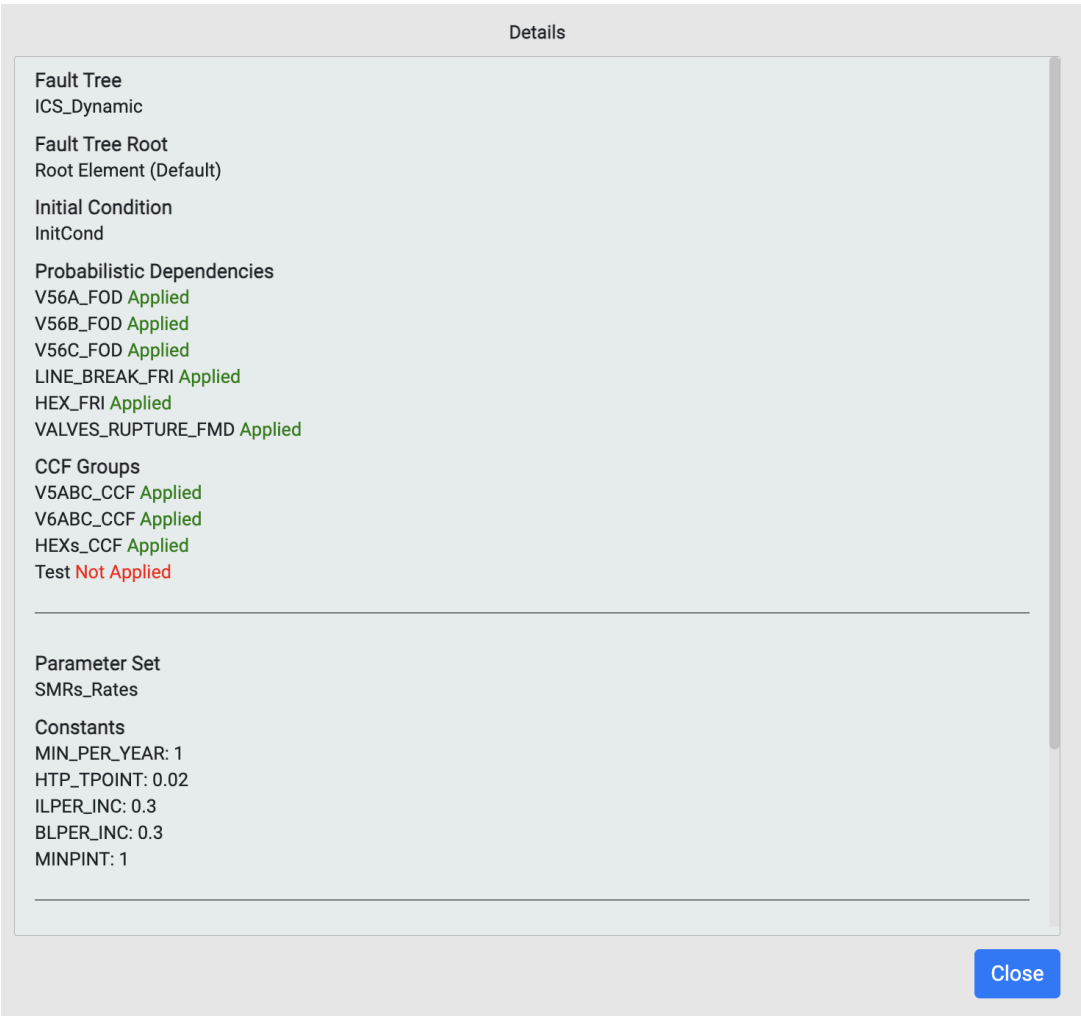
We provide an interface to plot and compare measures of interest e.g. reliability, MTTF, etc. against different parameters of interest. Click on the “Graphs” link under “Computing” in the left panel and then click “Reliability”. The following window will appear:



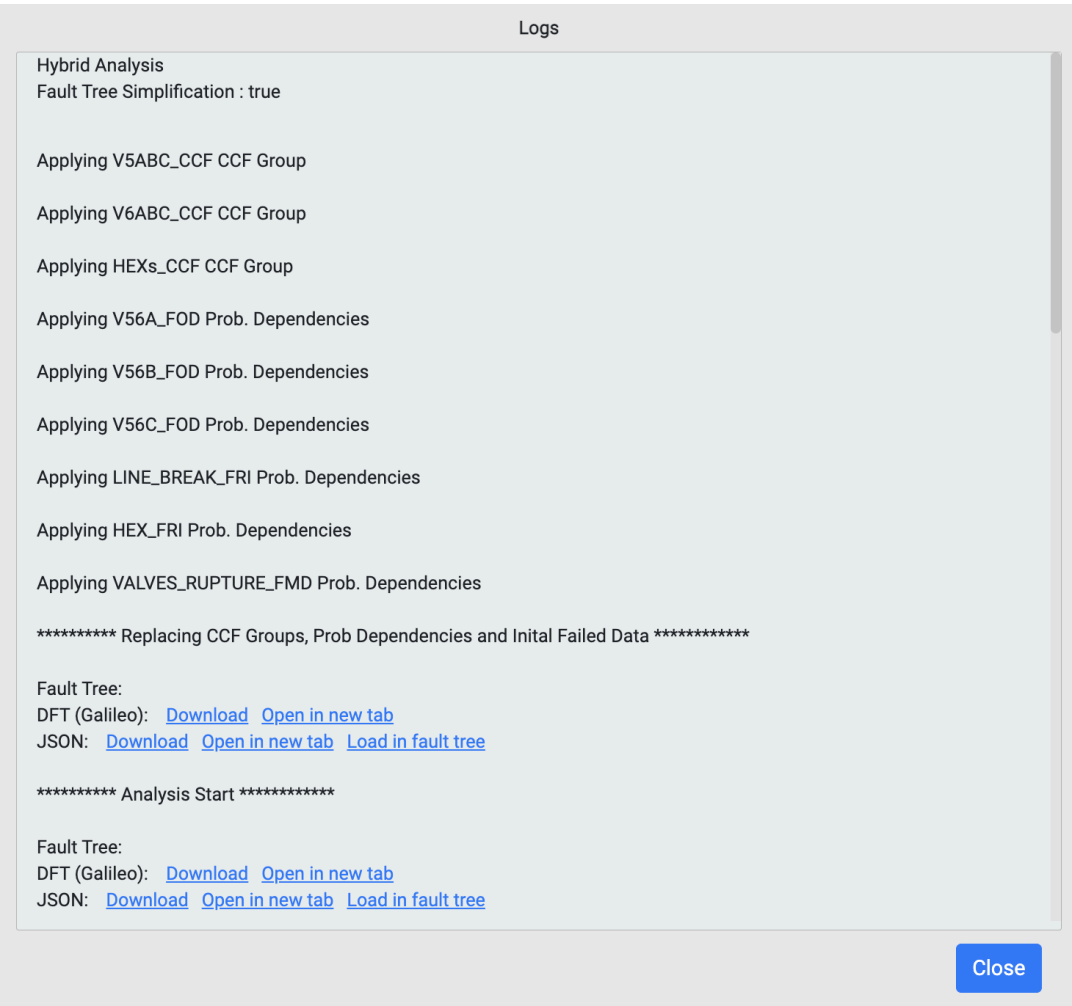
- All fields are filled up as described in the “Analysis” case with a few additions:
- One can specify a range of values of metric parameters as well as of Constants defined in the selected parameter set.
- A graph can be plotted on an existing graph as well that has the same variable on the X-axis.
- The variable on the X-axis of the graph can be selected either from the metric parameters or from the Constans of the selected parameter set.
- Click the “Start” button to display the graph:



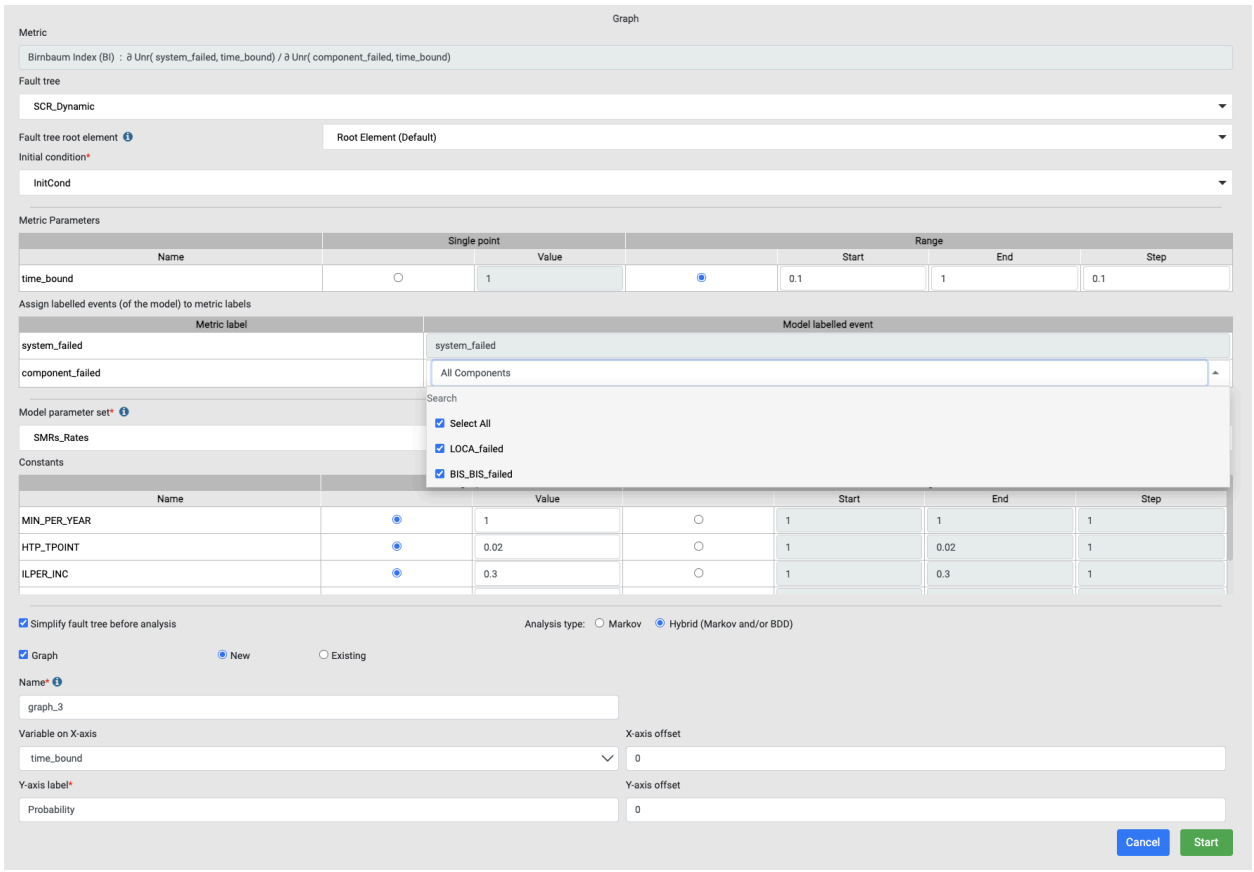
- Click the icon to rerun the analysis and draw a graph.
- Click the icon to stop the running analysis.
- Click the icon to view the configuration of the analysis.

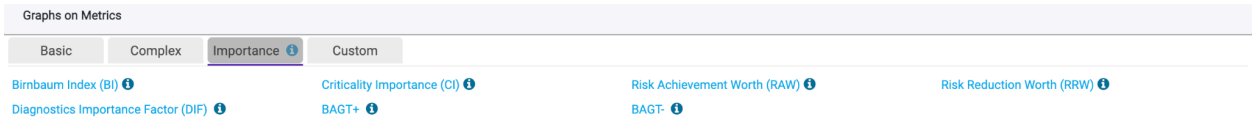


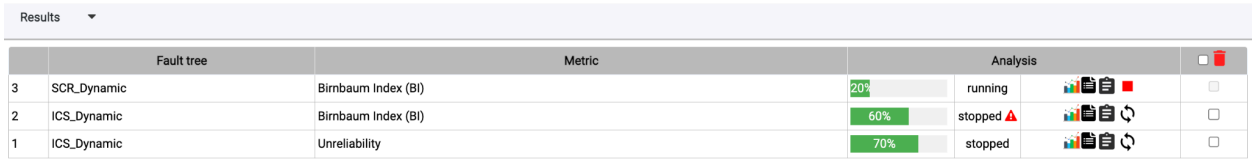
- Click the icon to view the analysis log.

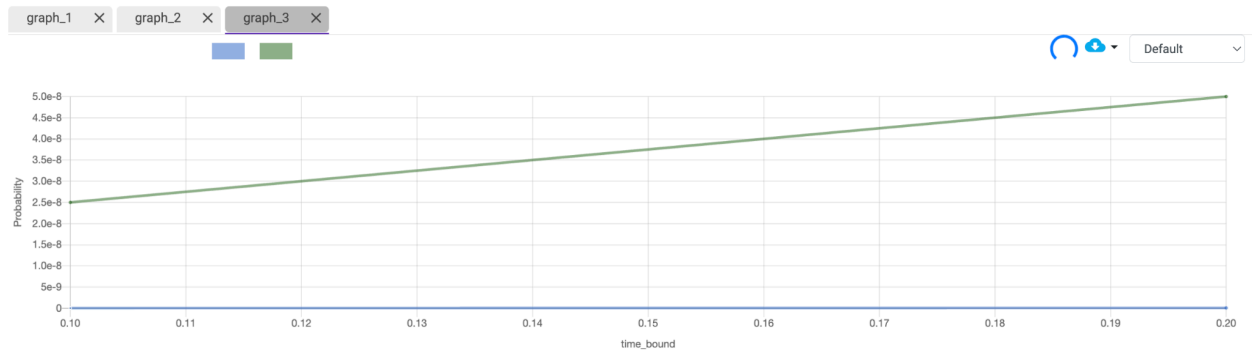


  - Click the “Download” link to download the generated artifact (DFT/DRN).
  - Click the “Open in new tab” link to open the generated artifact in a new browser tab.
  - Click the “Load in fault tree” link to load the generated fault tree in the current project.
- In the case of Importance measures, one can draw a plot for multiple components at the same time:


Graph
Metric
Birnbaum Index (BI) : ∂ Unr(system_failed, time_bound) / ∂ Unr(component_failed, time_bound)
Fault tree
SCR_Dynamic
Fault tree root element
Root Element (Default)
Initial condition*
InitCond
Metric Parameters
Single point
Range
Name
Value
Start
End
Step
time_bound
1
0.1
1
0.1
Assign labelled events (of the model) to metric labels
Metric label
Model labelled event
system_failed
system_failed
component_failed
All Components
Search
Select All
LOCA_failed
BIS_BIS_failed
Model parameter set*
SMRs_Rates
Constants
Name
Value
Start
End
Step
MIN_PER_YEAR
1
1
1
1
HTP_TPOINT
0.02
1
0.02
1
ILPER_INC
0.3
1
0.3
1
Simplify fault tree before analysis
Analysis type:
Markov
Hybrid (Markov and/or BDD)
Graph
New
Existing
Name*
graph_3
Variable on X-axis
time_bound
X-axis offset
0
Y-axis label*
Probability
Y-axis offset
0
Cancel
Start


Graphs on Metrics
Basic
Complex
Importance
Custom
Birnbaum Index (BI)
Criticality Importance (CI)
Risk Achievement Worth (RAW)
Risk Reduction Worth (RRW)
Diagnostics Importance Factor (DIF)
BAGT+
BAGT-

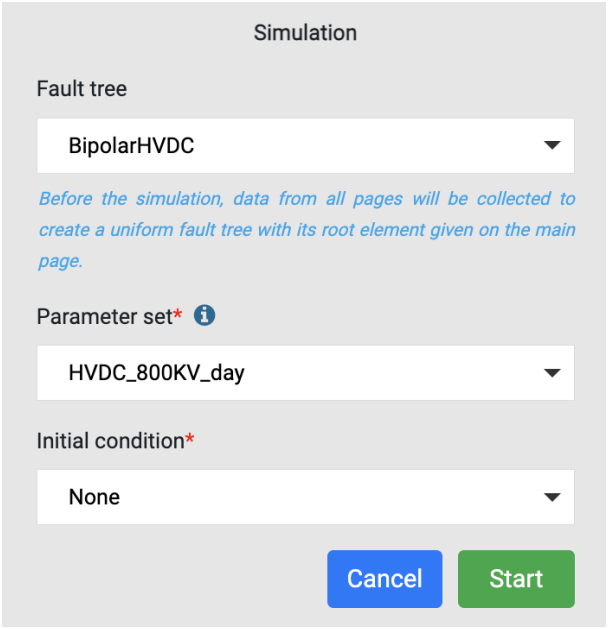

Results
Fault tree
Metric
Analysis
3
SCR_Dynamic
Birnbaum Index (BI)
20%
running
2
ICS_Dynamic
Birnbaum Index (BI)
60%
stopped
1
ICS_Dynamic
Unreliability
70%
stopped


graph_1
graph_2
graph_3
Default
5.0e-8
4.5e-8
4.0e-8
3.5e-8
3.0e-8
2.5e-8
2.0e-8
1.5e-8
1.0e-8
5e-9
0
Probability
0.10
0.11
0.12
0.13
0.14
0.15
0.16
0.17
0.18
0.19
0.20
time_bound

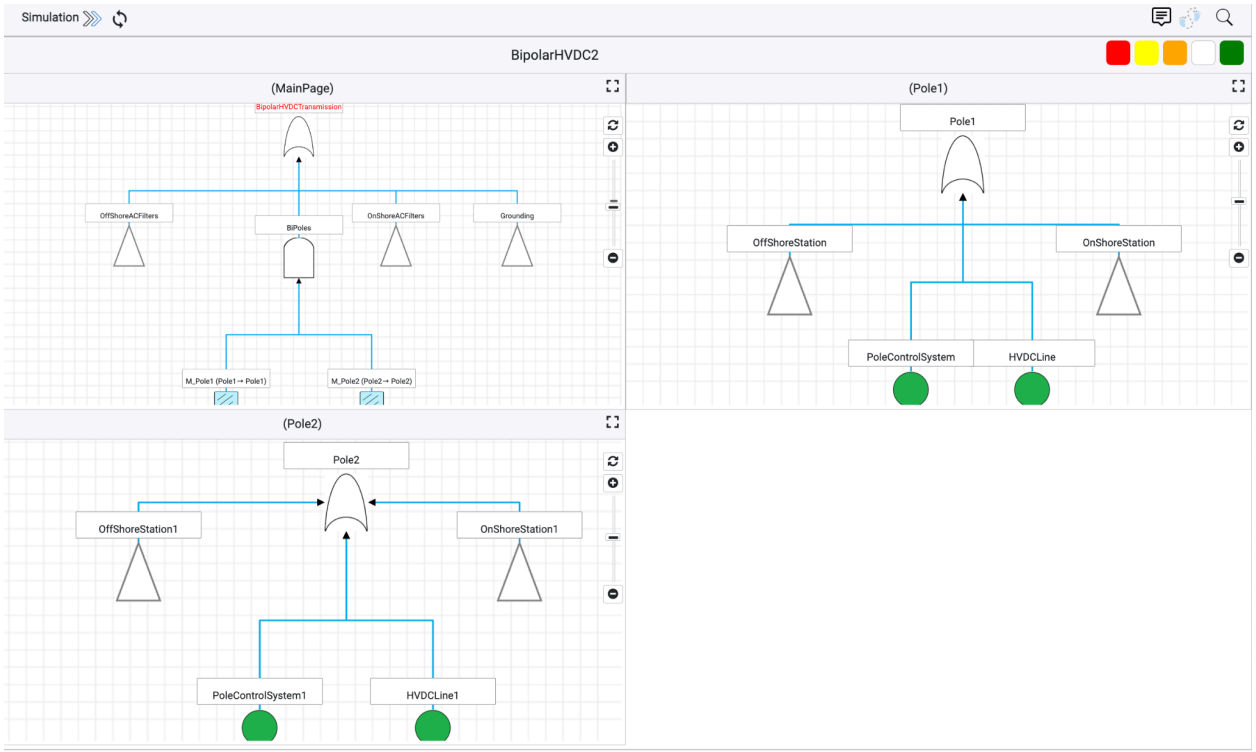
Interactive simulation

The idea is to interactively visualize a sequence of failures in a fault tree. The user would start with a usual fault tree and could select one of the basic events (BE) that should fail first. Based on this, the status of each element (failed, operational, fail-safe, claiming in SPAREs, etc.) is redetermined and then visualized. Afterward, another BE can be selected to fail, and so forth. The main benefit of this feature is that the semantics of DFTs become much clearer as users can try out the behavior by themselves.

Click on the “Interactive Simulation” link under “Computing” in the left tab. The following screen will appear. Click on the icon  to start the simulation. The user will be prompted to select a fault tree, a parameter set, and an initial condition. Note that if an initial condition is selected, the resultant fault tree will be shown on a single page because of CCF groups and probabilistic dependencies among elements that lie on multiple pages.



- Click the icon to start the simulation. The user will be prompted to select BEs, having constant probability distribution, that will be failed on start. On clicking the “Ok” button, the simulation will start.



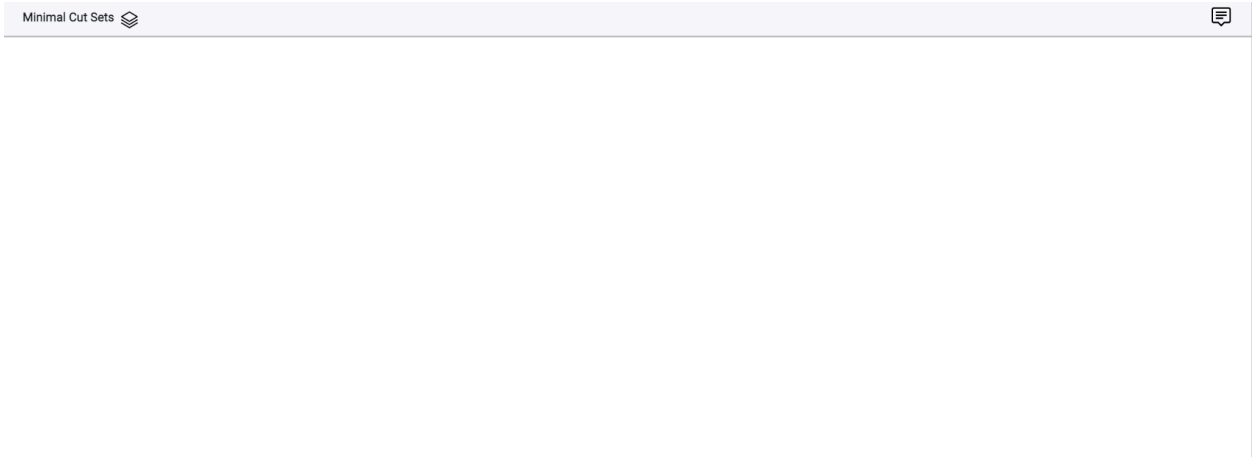
- All basic events (BEs) that can fail are shown in green.
- The user clicks any green BE to fail it. Its color will be turned into Red. After this, BEs that are operational and cannot fail currently remain White, those that are in a fail-safe state are Orange, and those that are in a dont-care state are Yellow.
- The user keeps on failing green BEs, and in return, the failure keeps on moving up the tree until the top-level event turns Red showing the failure of the top-level event.
- The sequence of failures can be shown by clicking on the icon  :

- Users can restart the simulation by clicking on the icon .

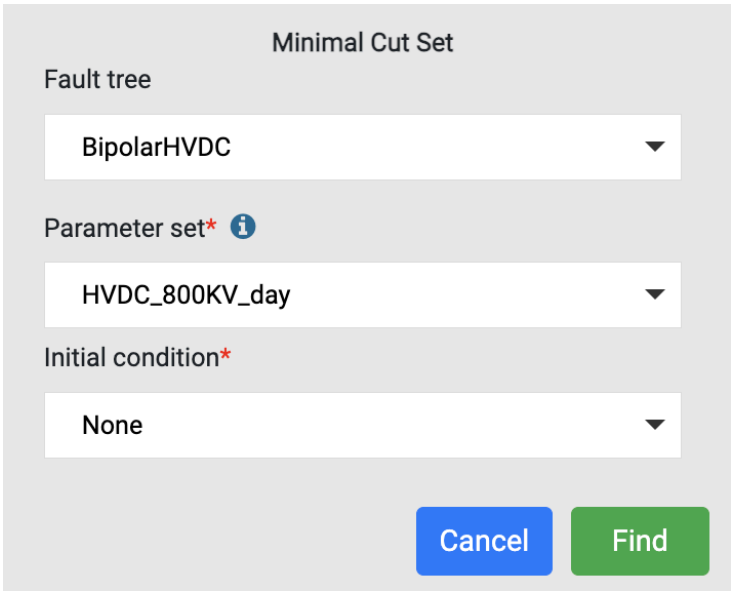
Minimal cut set (for static fault trees)

Cut sets represent sets of BEs whose failure leads to the failure of the top-level element of a fault tree. A minimal cut set is a set whose proper subset cannot be a cut set itself. Cut sets cannot be calculated for dynamic fault trees because of the dynamic nature of the system.

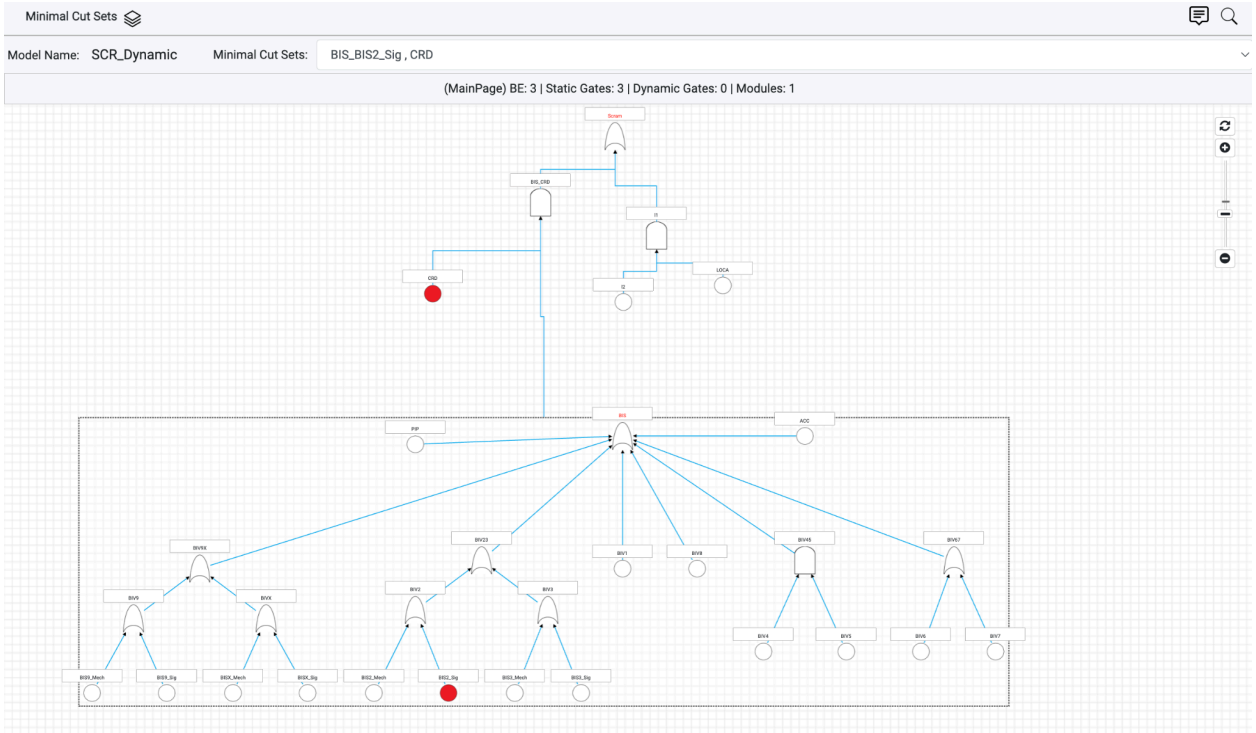
Click on the “Minimal cut set” link under “Computing” in the left tab. The following screen will appear.



Click on the icon  to start. The user will be prompted to select a failure model and a parameter set as:



On clicking “Find”, minimal cut sets are computed and displayed on the screen as:



- All minimal cut sets will be shown in the dropdown.

- On clicking a cut set, the corresponding BEs will be highlighted (in Red) in the tree.

Event Tree Analysis

It contains all event trees along with their configurations (parameter sets, loss sets, consequence sets), and computing methods.

Configurations

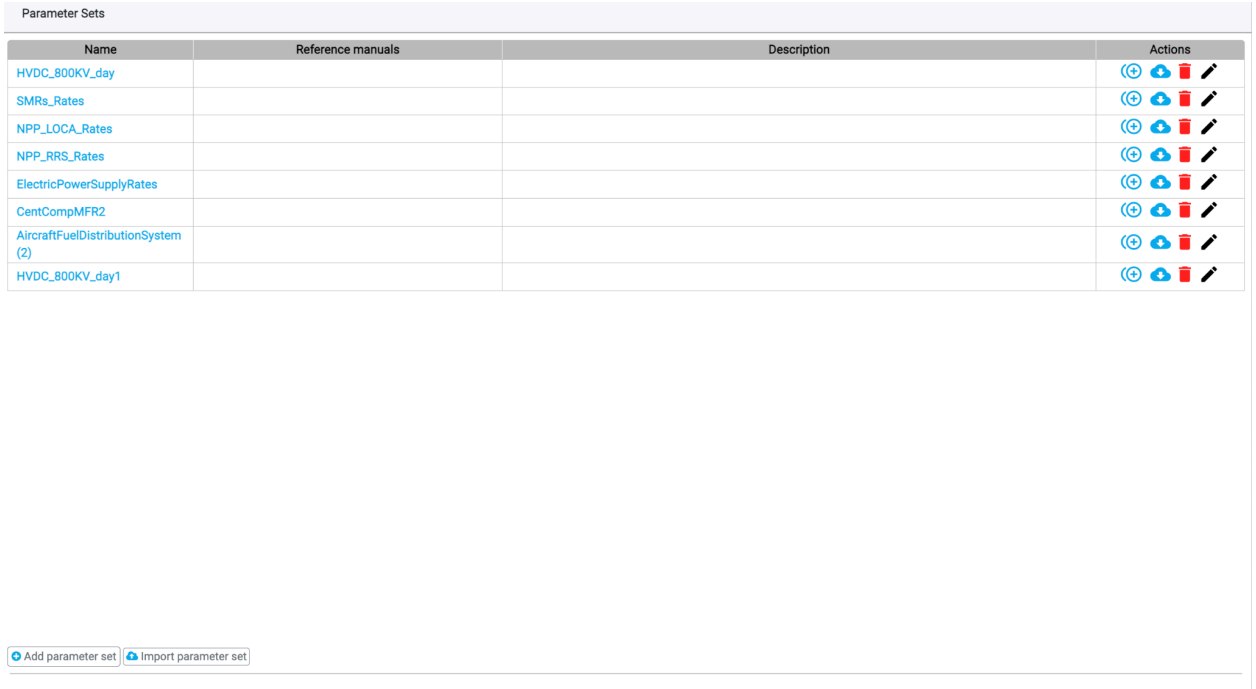
Parameter Sets

Each parameter set contains a list of parameters that are key-value pairs. They are used to specify different quantities in event trees e.g. probabilities, etc. By changing their values several variants of event trees can be generated, which can then be compared with each other based on the results of metrics of interest. Each parameter set comprises:

- Constants
- Constant expressions

View

Click the “Parameter Sets” under “Configurations” in the left panel to view all existing parameter sets.



Creation

Click the “Add parameter set” button to create a new parameter.

Import

Click the “Import parameter set” to import the parameter set in .ps format (a format in which parameter sets are imported/exported in SAFEST).

Duplicate

Click the icon to duplicate a parameter set.

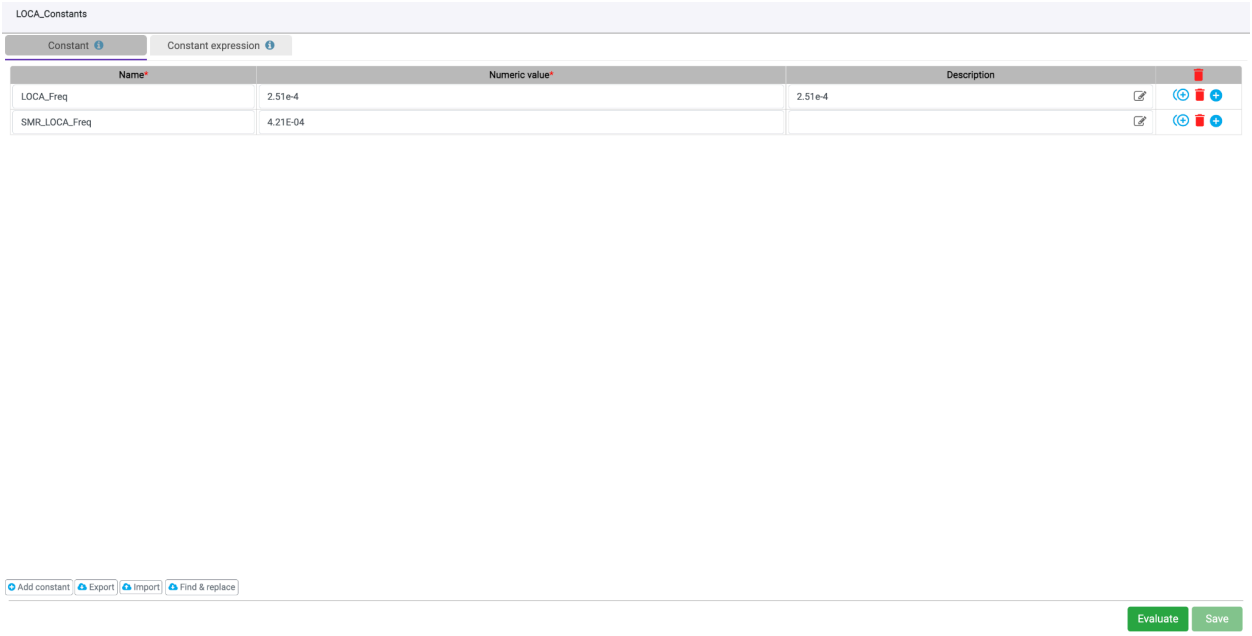
Export

Click the icon to export a parameter set in .ps format.

Update

Click a parameter set to update its details

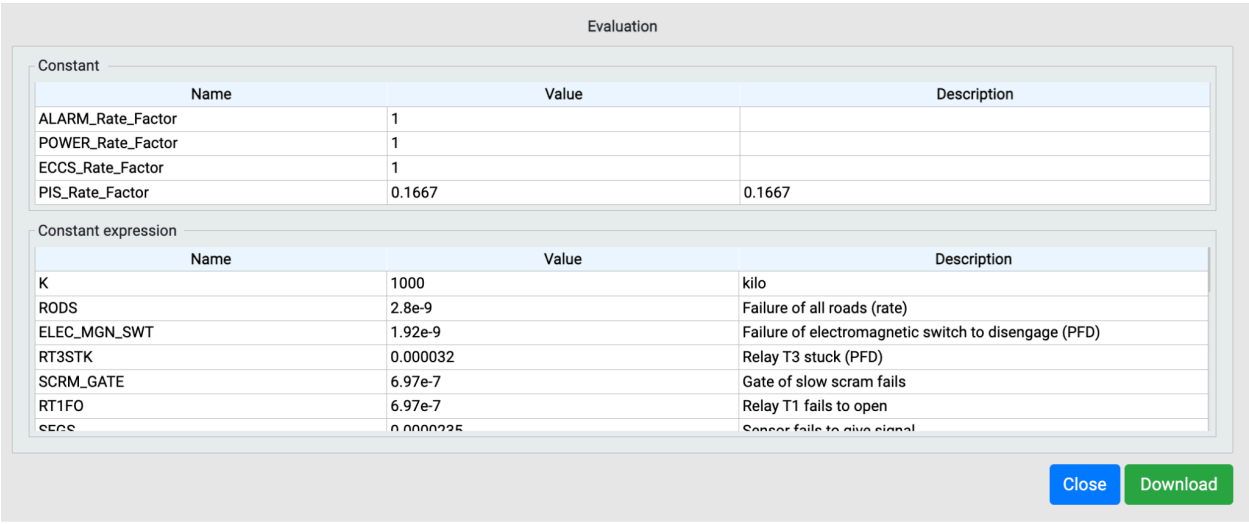
Constants



Constants can only be numeric e.g. 4, 2.3, 4e-6 etc. Their value can be changed at the time of analysis. For example, graphs can be plotted for matric results against ranges of values of constants.

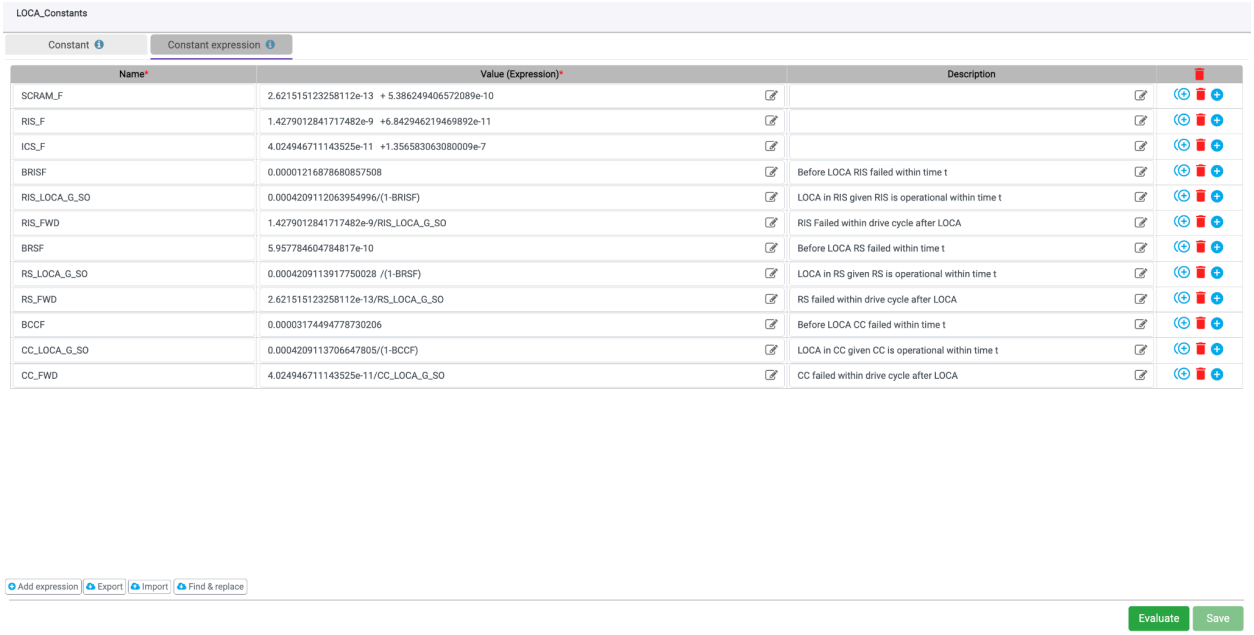
- Click the “Add constant” button to enter a new row in the table.
- Click the “Export” button to export constants in a CSV format.
- Click the “Import” button to import constants from a CSV file.
- Click the “Find & replace” button to search sub-strings among Constants and replace them with a new string.

- Click the ⊕ icon in the last column of any row to add a new row above it.
- Click the ⊕ icon to duplicate the row.
- Click “Evaluate” button to evaluate expressions in all tabs. The result will be displayed in a popup.



- Click the “Save” button to save the data. This action will save data in all the tabs.

Constant Expressions

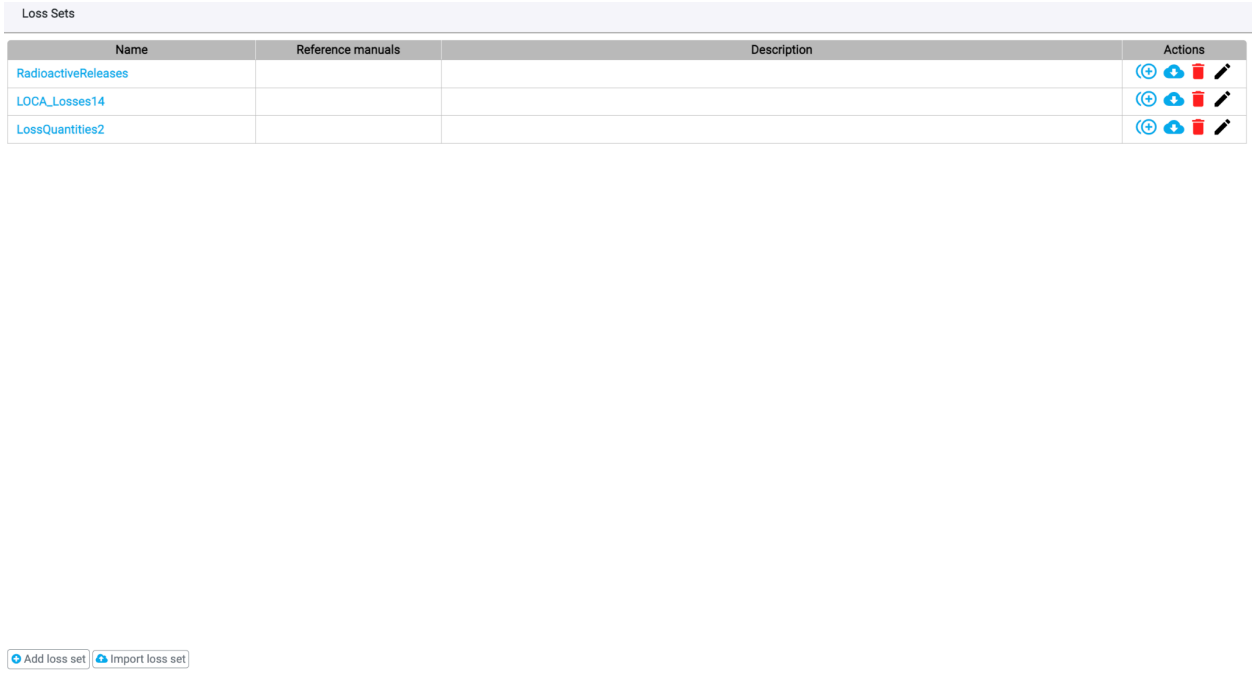


These are non-negative, real-value expressions, which can use constants (defined in Constants tab) e.g. x + 2 where x is a constant. The grammar of the expression is given here.

- Click the “Add expression” button to enter a new row in the table.
- Click the “Export” button to export expressions in a CSV format.
- Click the “Import” button to import expressions from a CSV file.
- Click the “Find & replace” button to search sub-strings among Constant Expressions and replace them with a new string.

- Click the  icon to duplicate the row.
- Click the  icon in the last column of any row to add a new row above it.
- Click the “Save” button to save the data. This action will save data in all the tabs.

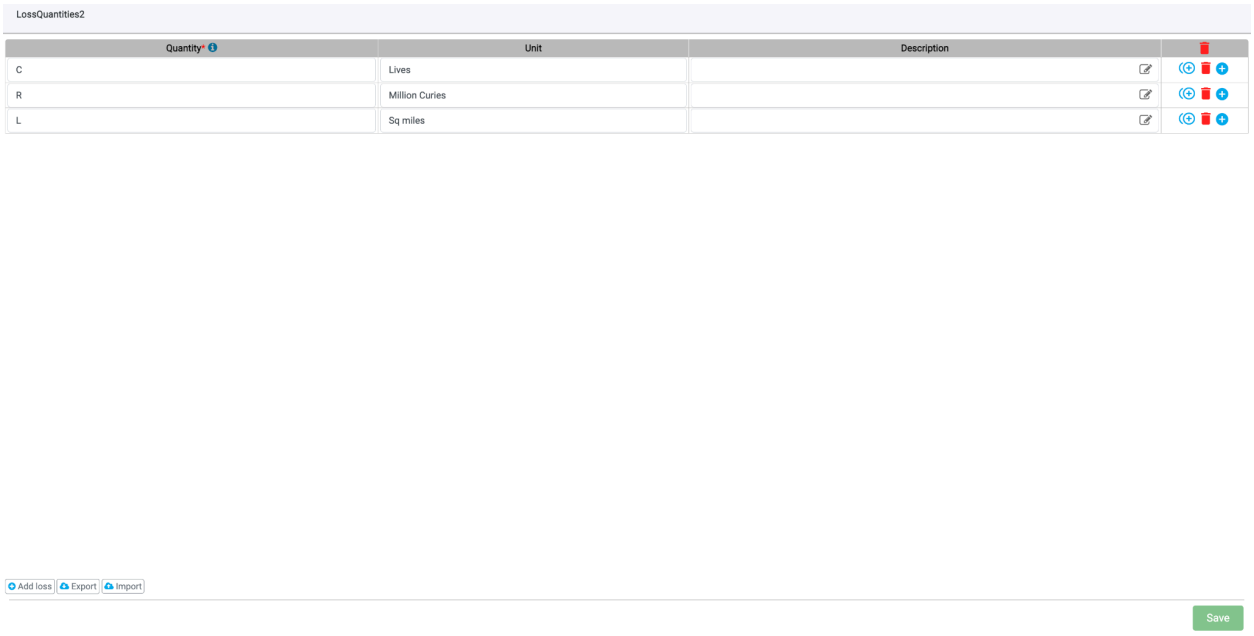
Loss Sets



A loss set contains quantities along with their units and descriptions. These quantities can be assigned values at nodes/states of event trees. During analysis, their expected values are calculated by the analysis algorithms.

Click the “Loss Sets” in the left panel to view loss sets.

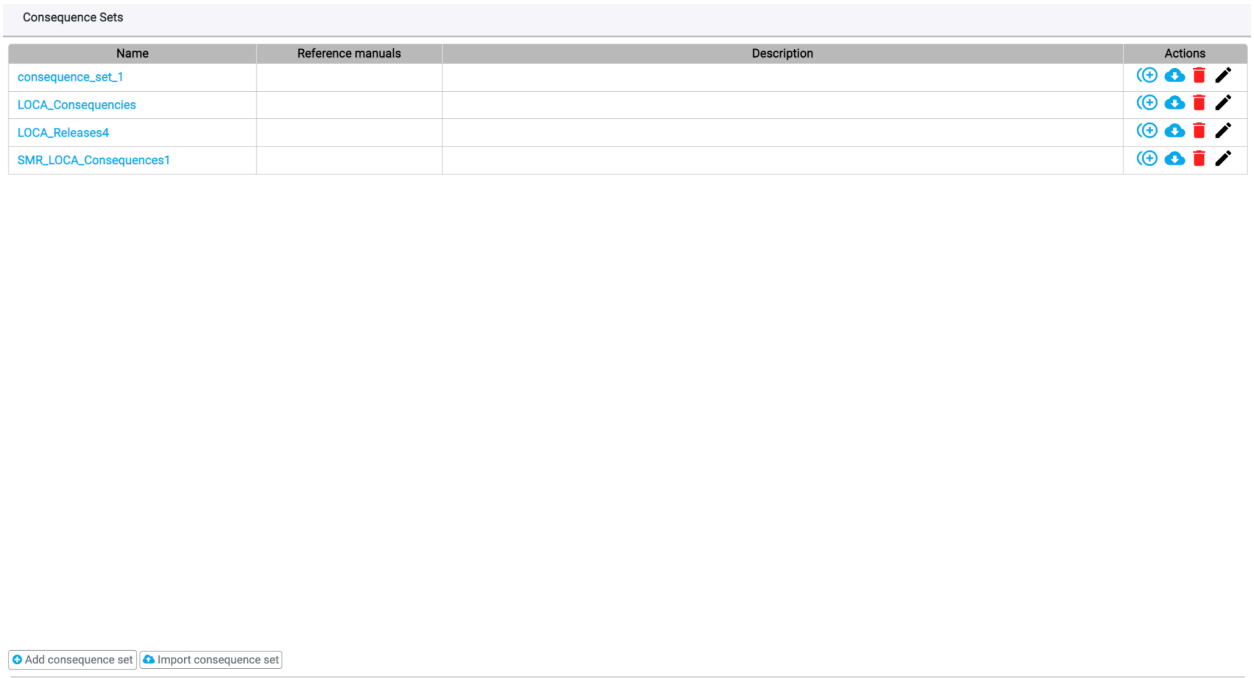
- Click the “Add loss set” button to create a new loss set.
- Click the “Import loss set” button to import the loss set in .rs format (a format in which loss sets are imported/exported in our tool).
- Click the icon to duplicate the row.
- Click the icon to export a loss set in .rs format.
- Click a loss set to view its details
    - Click the “Add loss” button to add a new loss quantity in the above table.
    - Click the “Export” button to export loss in a CSV format.
    - Click the “Import” buton to import consequences in a CSV format.
    - Click the icon to duplicate the row.
    - Click the icon in the last column of any row to add a new row above it.
    - Click the “Save” button to save the data.



Consequence Sets

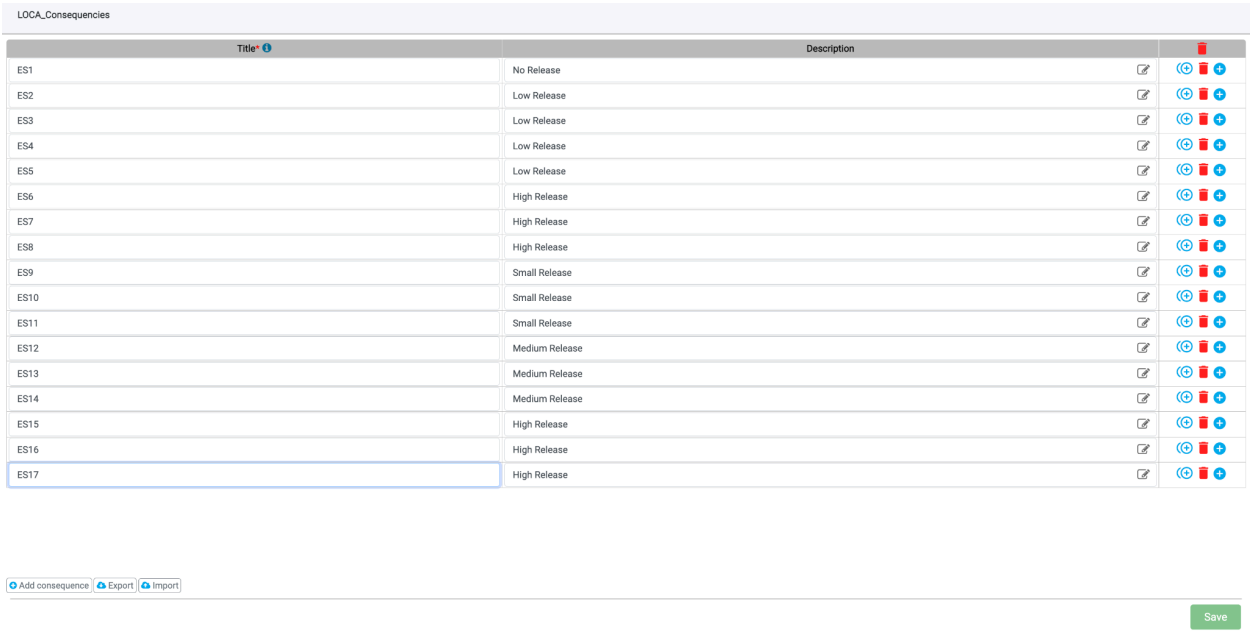
A consequence set contains a list of outcomes along with their descriptions. These outcomes can be assigned to leaf nodes/states in event trees. During analysis, their expected frequencies/probabilities are calculated.

Click the “Consequence Sets” in the left panel to view consequence sets.



- Click the “Add consequence set” button to create a new consequence set.

- Click the “Import consequence set” to import a consequence set in .cs format (a format in which loss sets are imported/exported in our tool).
- Click the icon to duplicate the row.
- Click the icon to export a consequence set in .cs format.
- Click a consequence set to view its details
    - Click the “Add consequence” button to add a new consequence in the above table.
    - Click the “Export/Import” button to export/import consequences in a CSV format.
    - Click the icon to duplicate the row.
    - Click the icon in the last column of any row to add a new row above it.
    - Click the “Save” button to save the data.

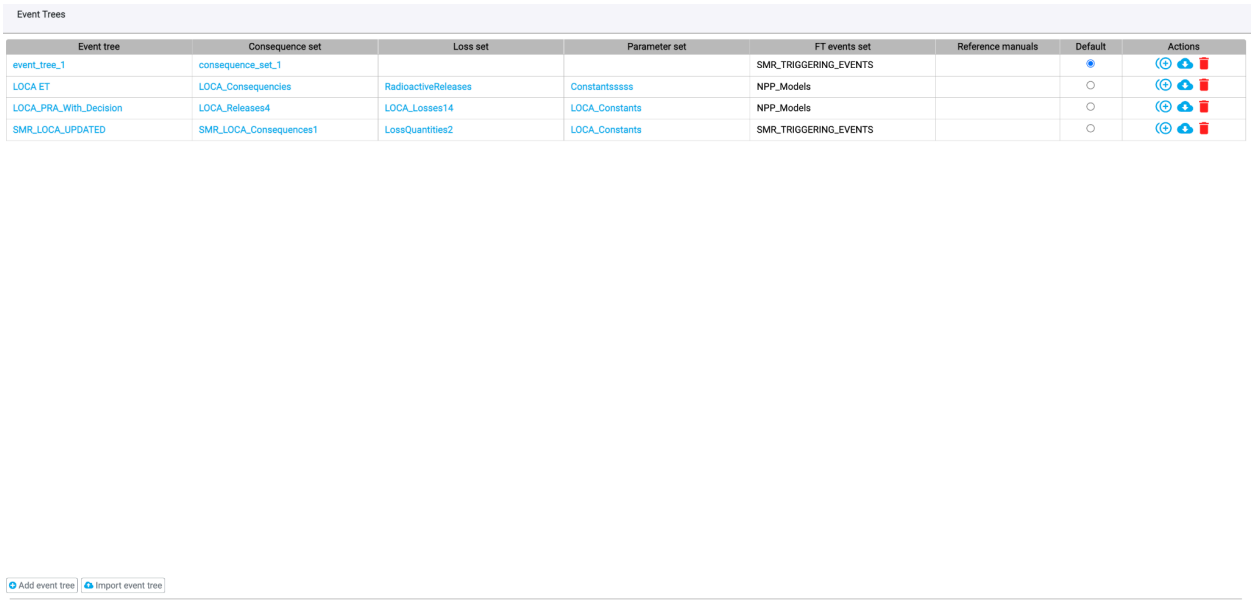


Event Trees

This section contains all event trees.

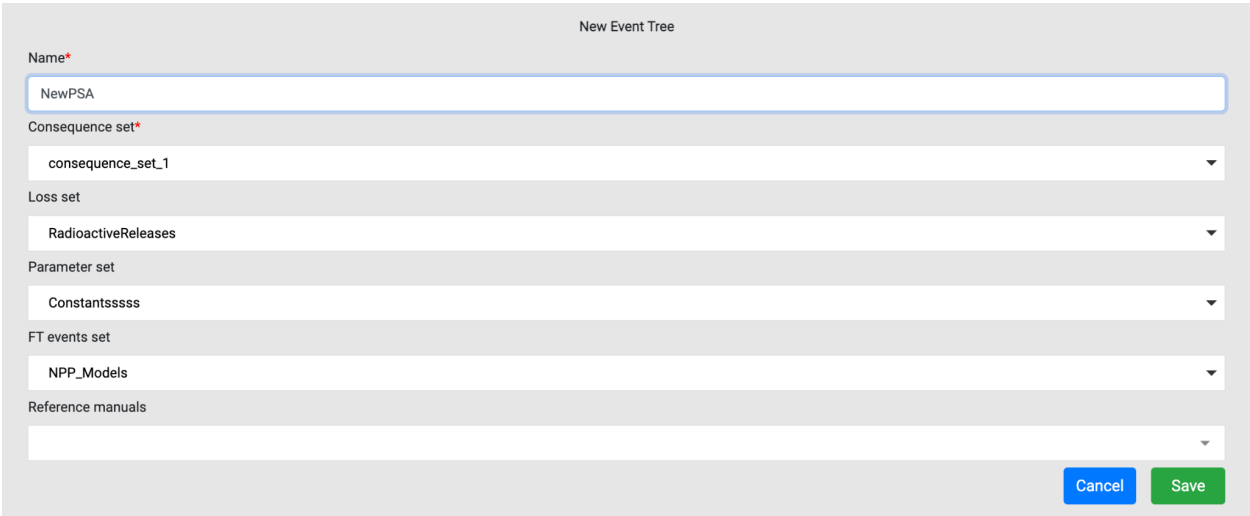
View

Click on the “Event Trees” in the left panel to display event trees in a table. A model that is worked upon the most can be selected as a default model by selecting the corresponding radio button.



Creation

Click the “Add even tree” to create a new event tree.



Enter all mandatory fields and click the “Save” button.

Export

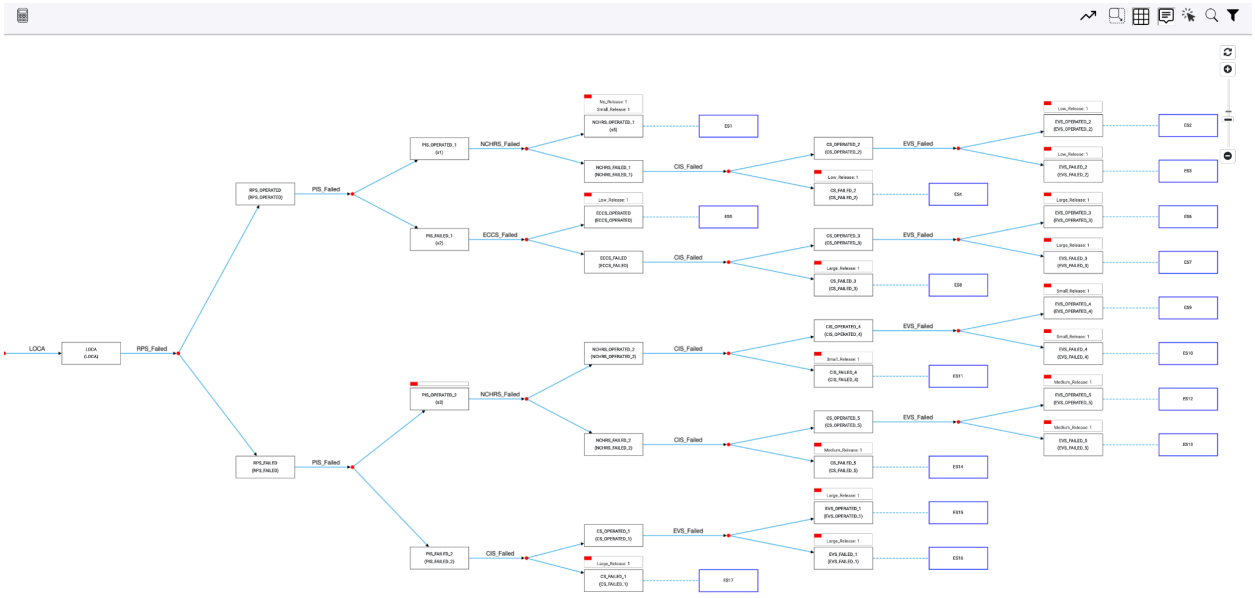
Click the icon in the corresponding row to download the event tree model in .et format (a format used in the SAFEST tool to import/export event tree models).

Import

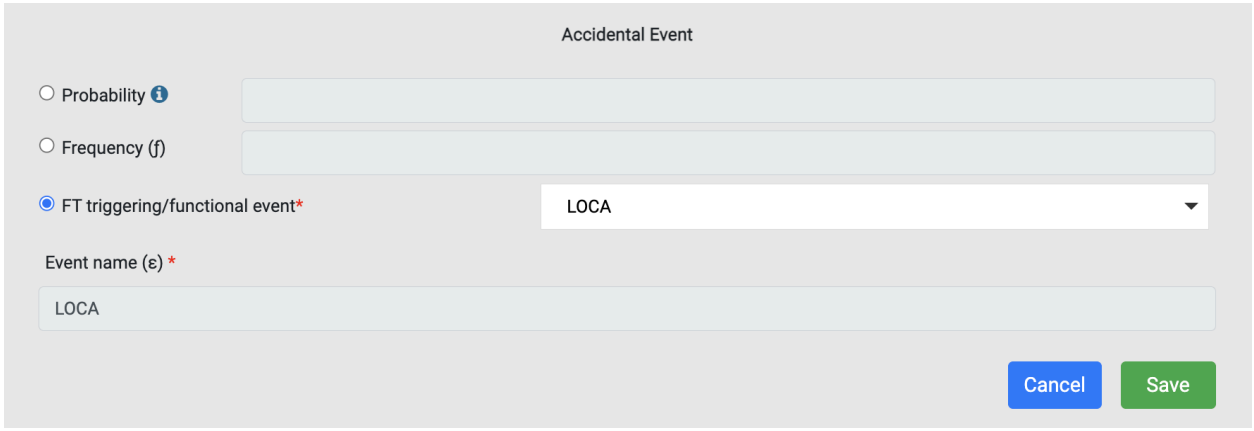
Click the “Import event tree” button to import an event tree model (saved in .et format) from your drive.

Update

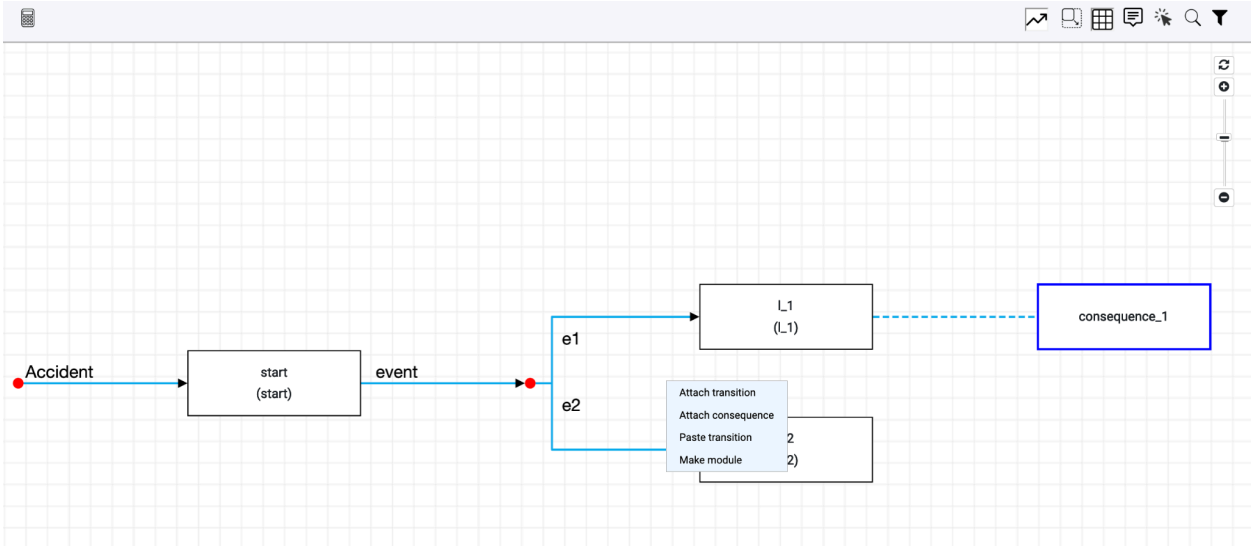
Click on the model in the table to open it in the canvas.



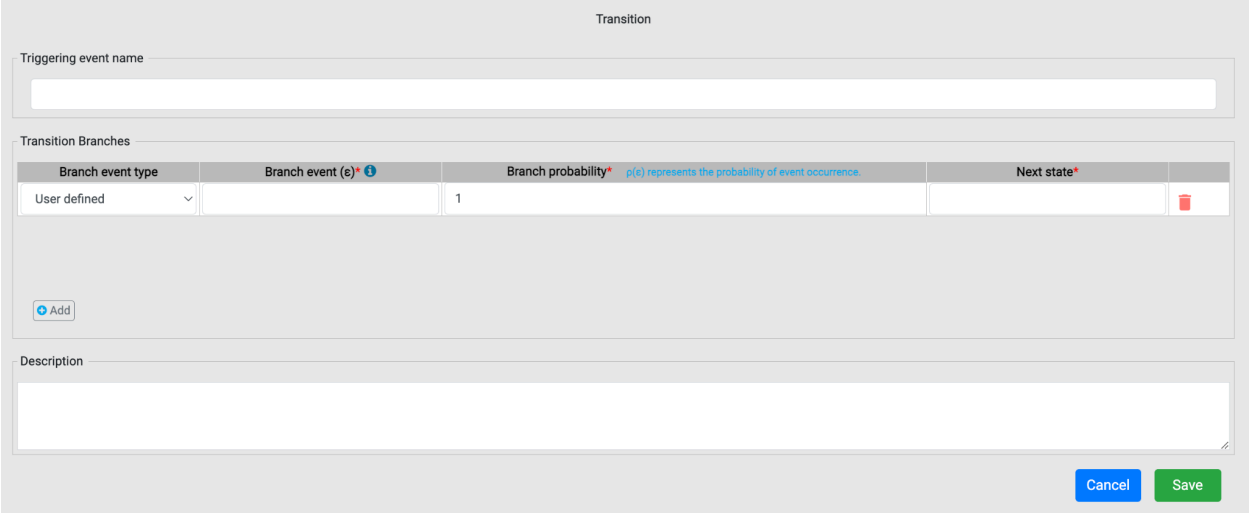
- Click on the first arrow to update the information of “Accidental Event” (initiating event).
  - For the Accidental Event, select the respective radio button to enter its:
    - Probability
    - Frequency
    - FT triggering/functional event – The dropdown will be filled with all FT events/FT event expressions that exist in the FT events sets associated with the event tree. Select an FT event from the dropdown menu as an accidental event.
  - Enter the name of the event in the “Event name” field.



- Right-click on a node to add a transition, a consequence, past a transition (which is already copied), or convert the sub-tree emanating from this node into a module.



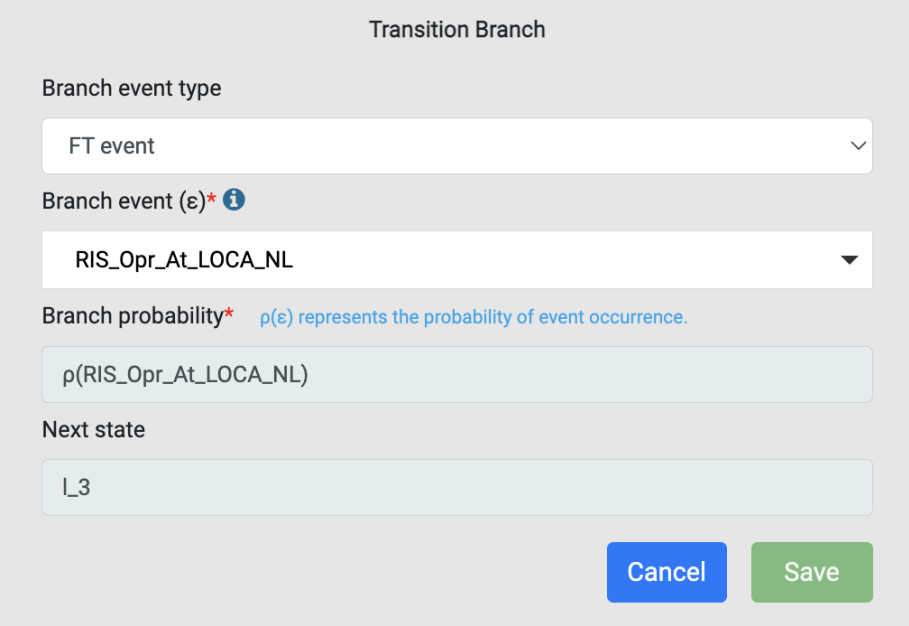
- Click the “Attach transition” to add a new transition to the selected node. You can add multiple transitions to a given node. Note that no further transition can be attached to a node having a consequence attached to it.



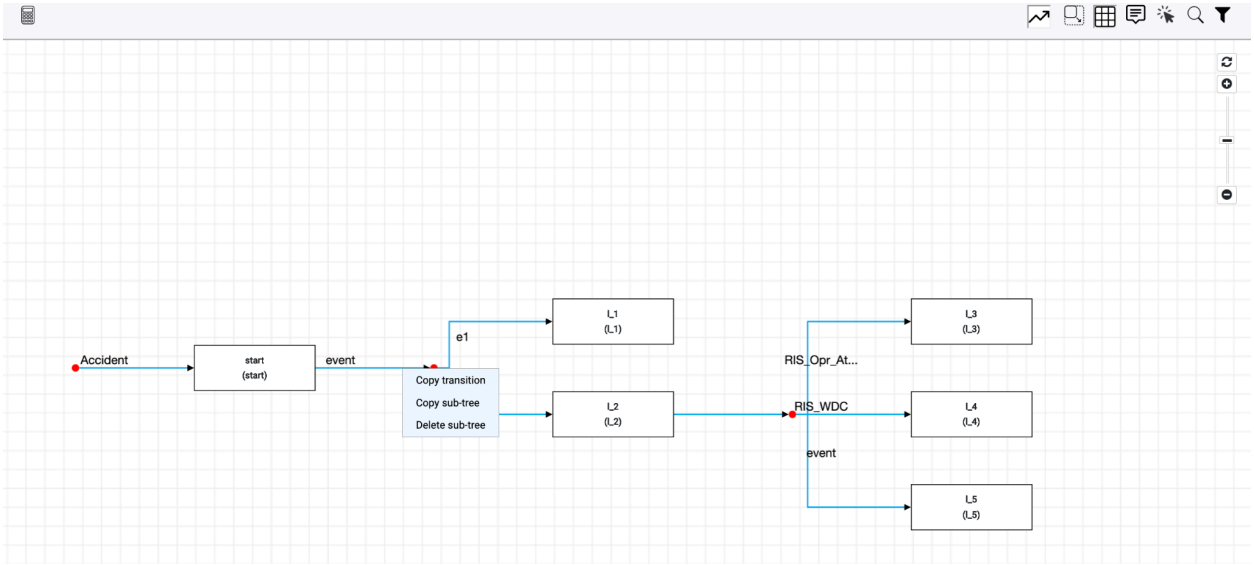
  - Enter the event name that triggers the branch: triggering event name
  - Click the “Add” button to add a new branch of the transition. One can add multiple branches but the sum of probabilities of all branches must be equal to 1.
  - Select the branch event type that triggers the branch: User-defined or FT event branch.
    - User-defined branch
      - Enter event name, branch probability – which can be a constant expression from the associated parameter set – and next state name
    - FT event brach

- Select FT event from the “Branch event” dropdown. The branch probability is the probability of the selected event that will be calculated before analyzing the event tree.

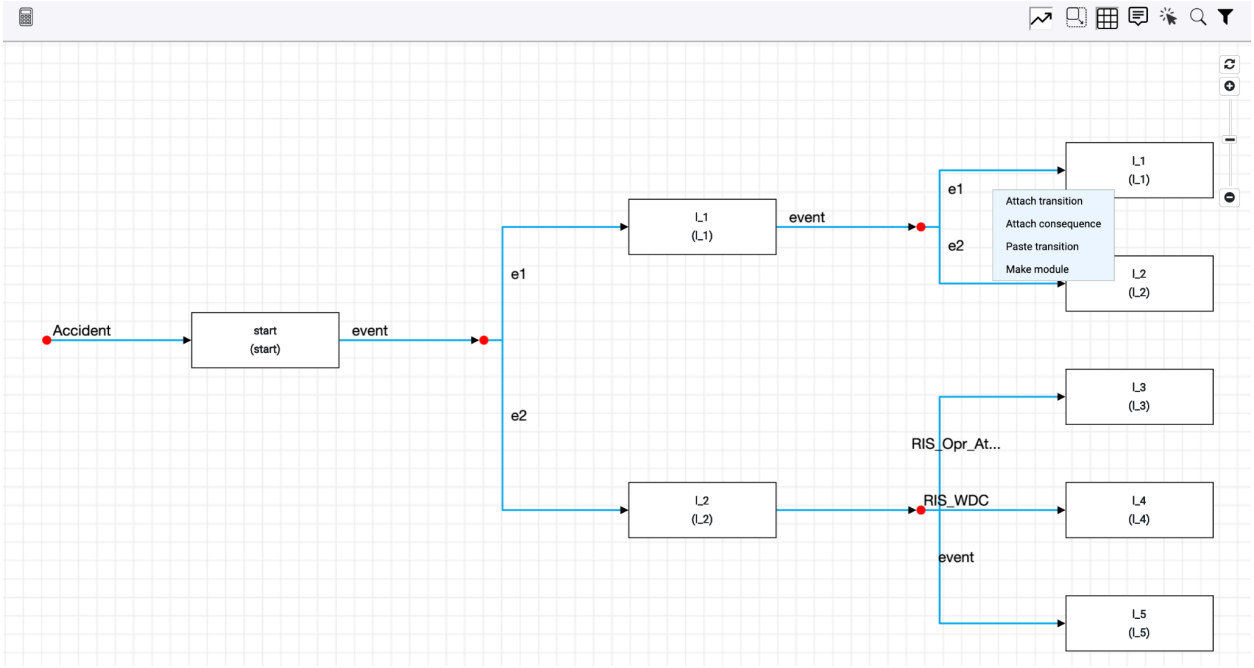
- Click a transition to update it. The popup, which comes up while adding a new transition, will appear.
- Click on a transition branch to update it.



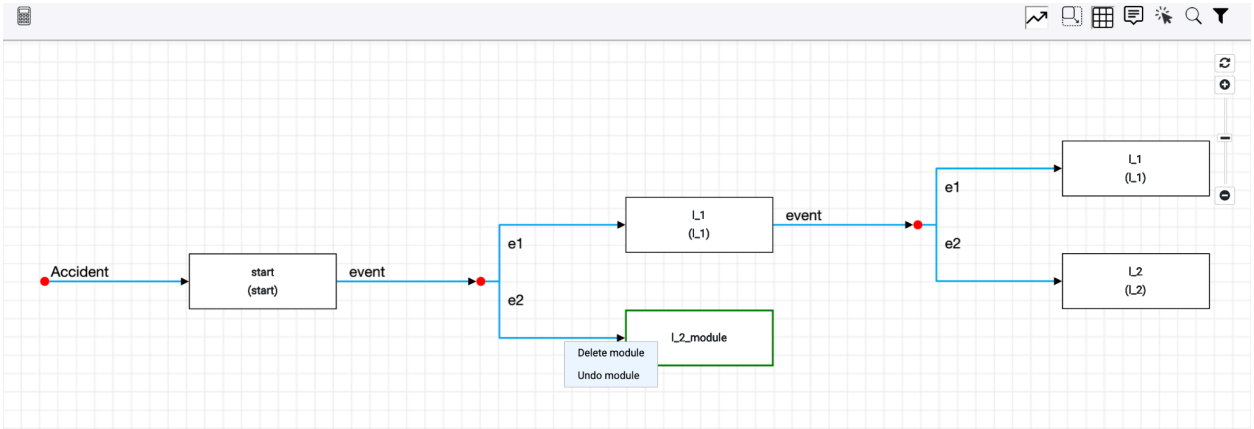
- Right-click on a transition to copy/delete a sub-tree originating from the transition, or copy the transition (along with its branches).



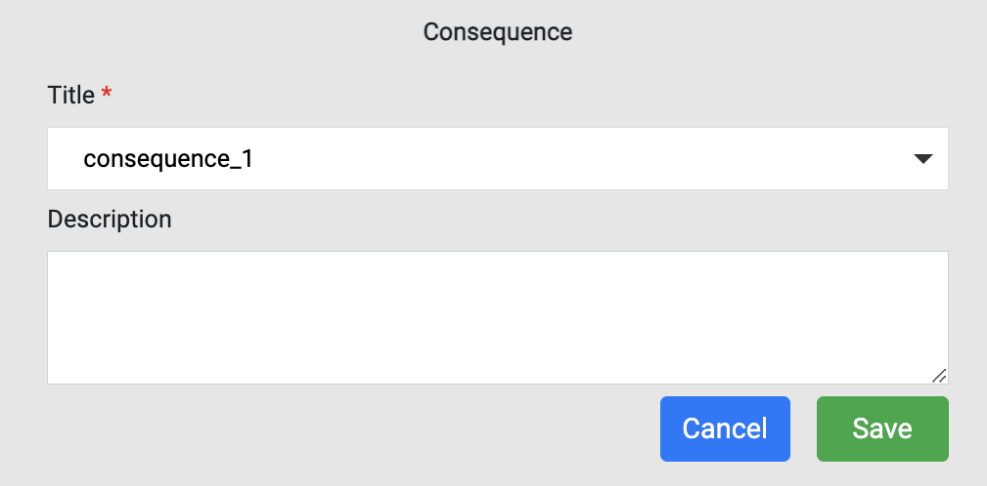
- Copy a transition/sub-tree and then right-click on a node to paste it. Note that transition branches cannot be copy-pasted individually.



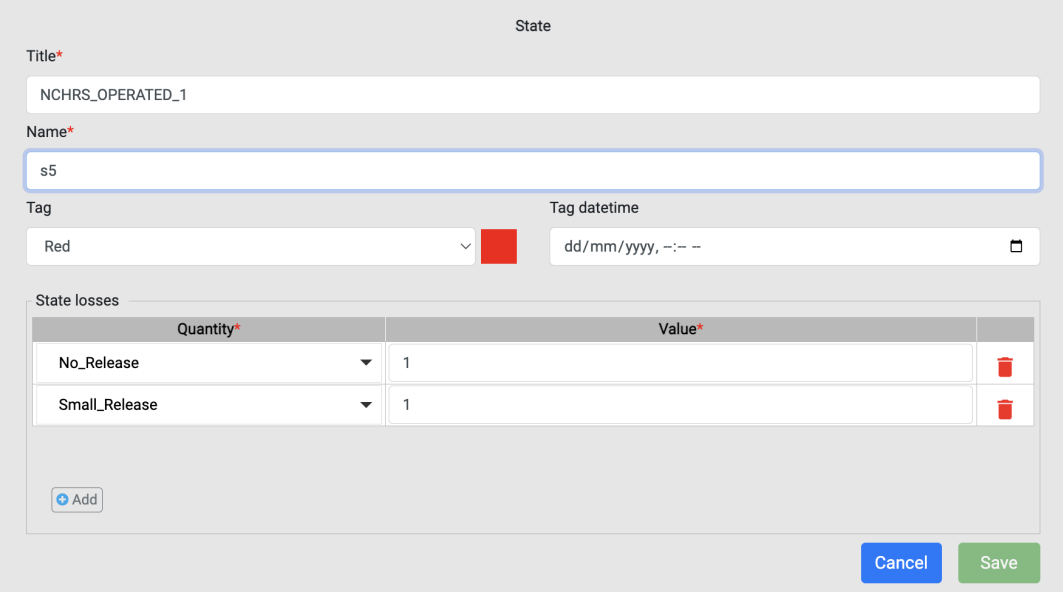
- Click on “Make module” to convert the sub-tree emanating from the node into a module. This helps simplify the event tree.
    - Undo a module by right-clicking a module node and selecting “Undo module”.
    - Delete a module by right-clicking a module node and selecting “Delete module”.



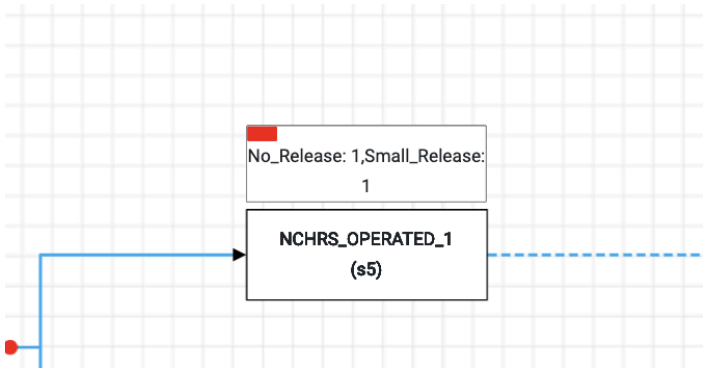
- Click the “Attach consequence” to attach a consequence with a node. All consequences in the “Consequence Set” attached to the model will be available in the “Title” dropdown. Select one of the consequences and click the “Save” button. Note that a consequence can be attached to the leaf nodes of a tree.



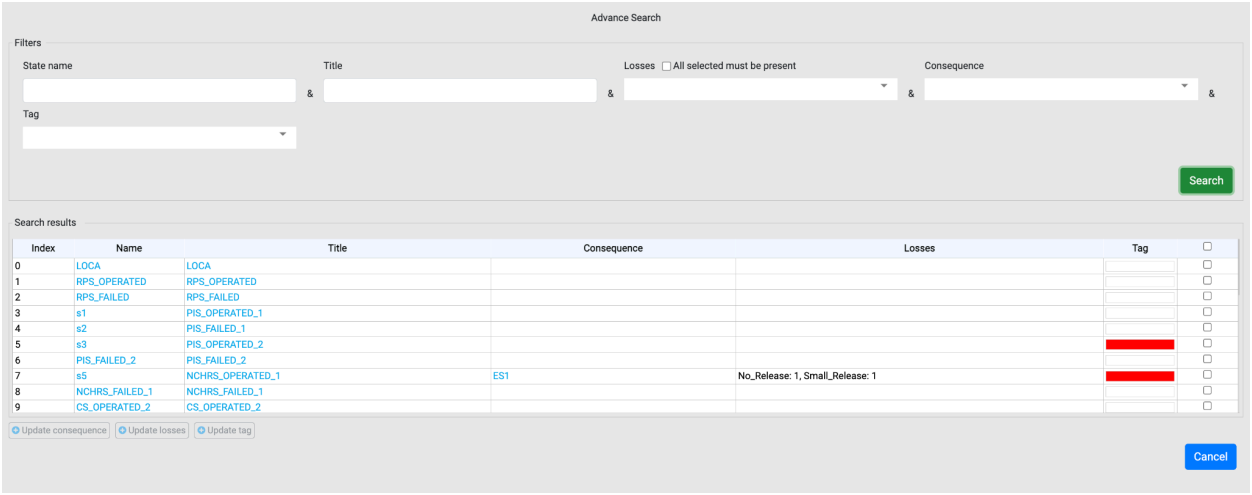
- Click a node to update it.
  - Update the title of the node (not unique in the event tree)
  - Update the name of the node (unique in the event tree)
  - Update tag color and tag datetime
  - Add losses along with their values (which can be constant expressions).



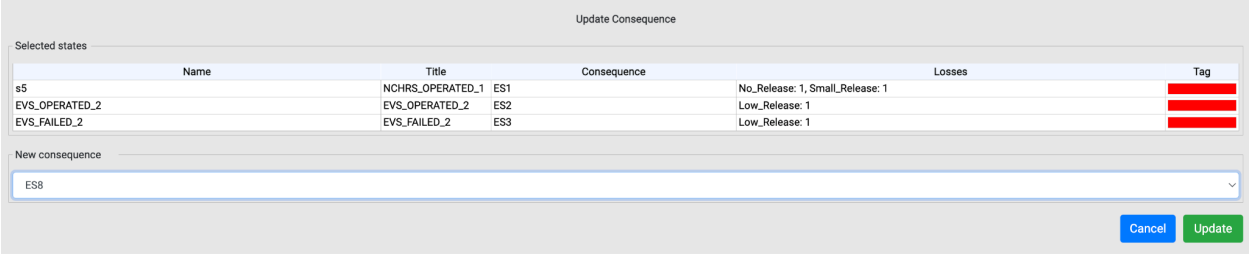
- Click on the icon  to enable the navigator at the bottom of the screen.
- Click on the icon  to show the grid on the screen.
- Click on the icon  to show the summary information about each element on the screen.



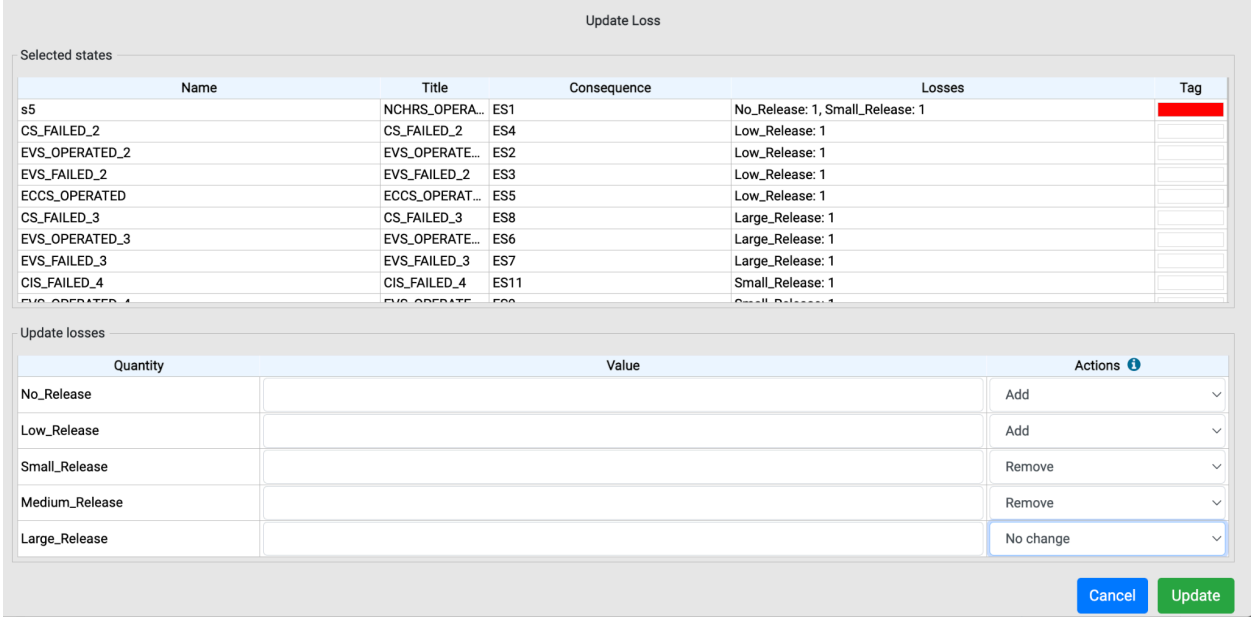
- Click on the icon  to display summary information about an element on hovering.
- Click on the icon  to search for any element on the screen.
- Click on the icon  to search elements based on filters. A popup will appear to apply filters.
    - All fields that are filled up, their data is AND together to find the nodes. Within a field, the data is OR-together unless it is mentioned otherwise.
    - If any file is not filled up, it is not considered to be a part of the filtering process, and thus ignored.
- On clicking the “Search” button, the result will be displayed in the “Search results” table.



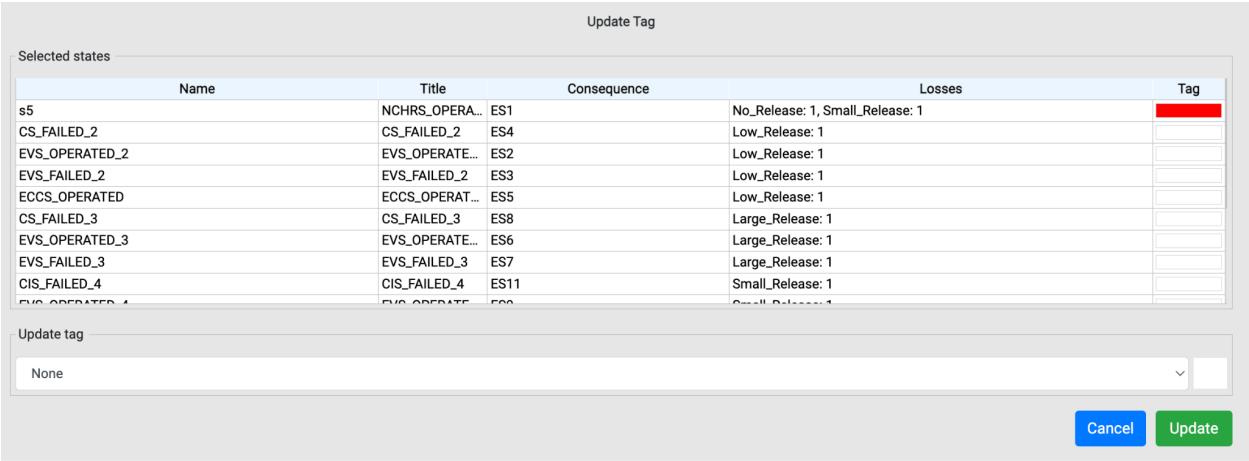
    - An element can be updated by clicking its name and opening the corresponding popup.
    - By selecting elements of the same type in the “Search results” table, operations can be performed on them at once.
        - Update consequences (for leaf nodes)



- Update losses



- Update tag



Computing

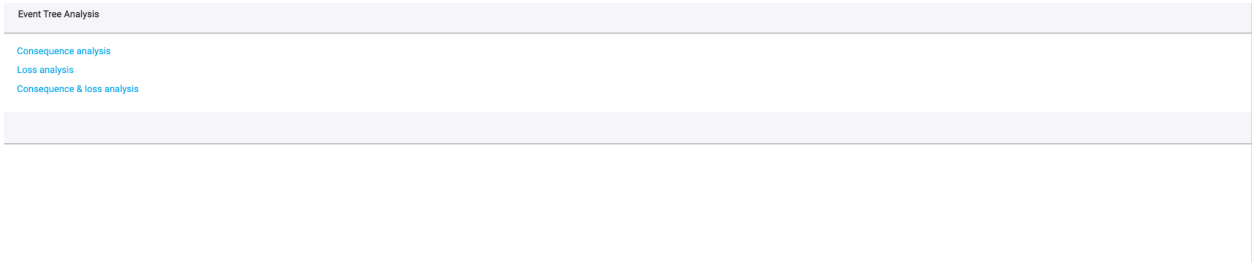
Event trees can be analyzed for consequence frequencies/probabilities and expected values of (loss) quantities.

- Analysis – the exact results of consequence frequencies/probabilities and expected values of (loss) quantities can be computed.

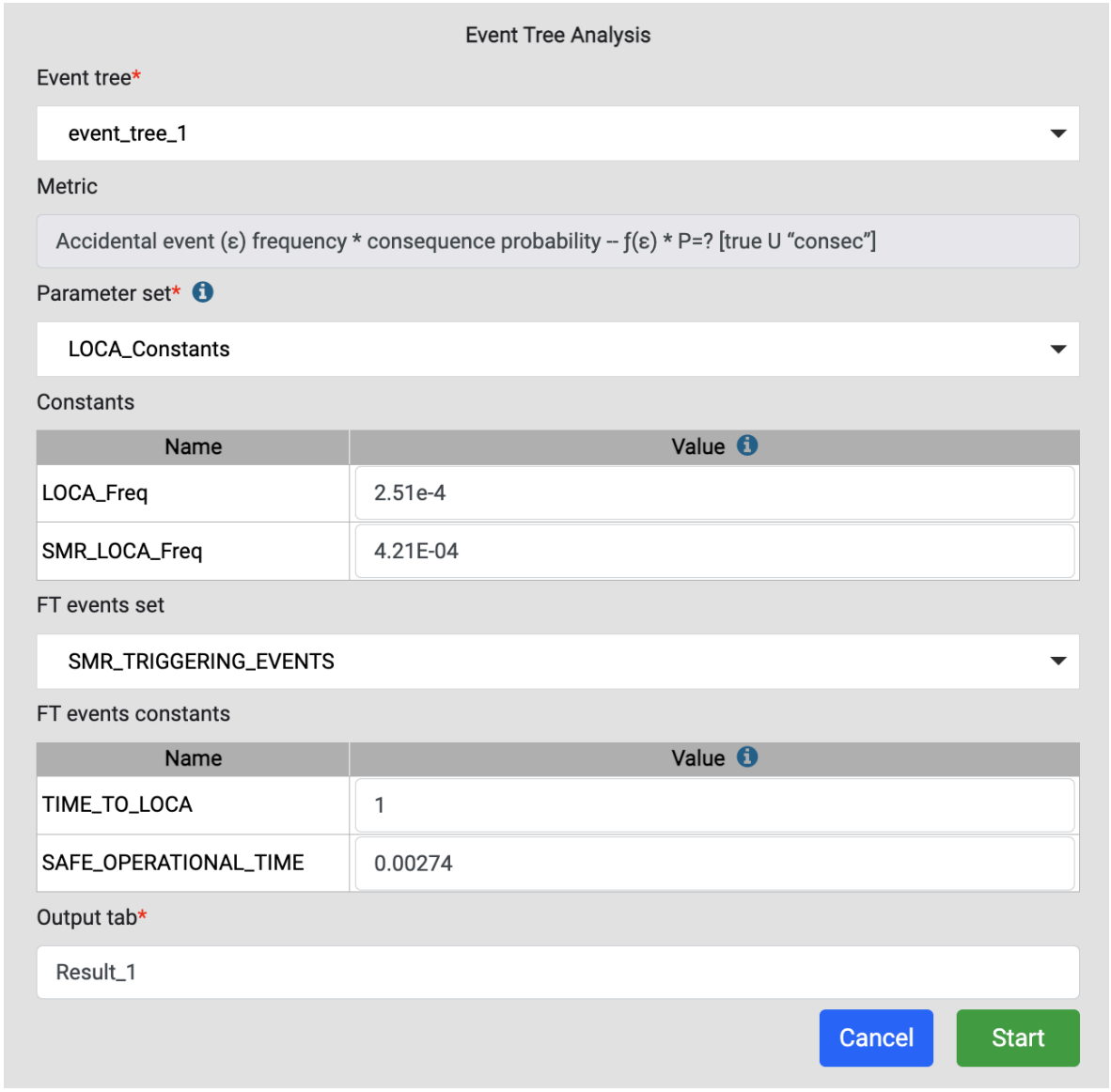
- Graphs – the exact results of consequence frequencies/probabilities and expected values of (loss) quantities can be graphed against time duration.

(Exact) Analysis

Click on the “Analysis” link under “Computing” in the left panel. The following window will appear.



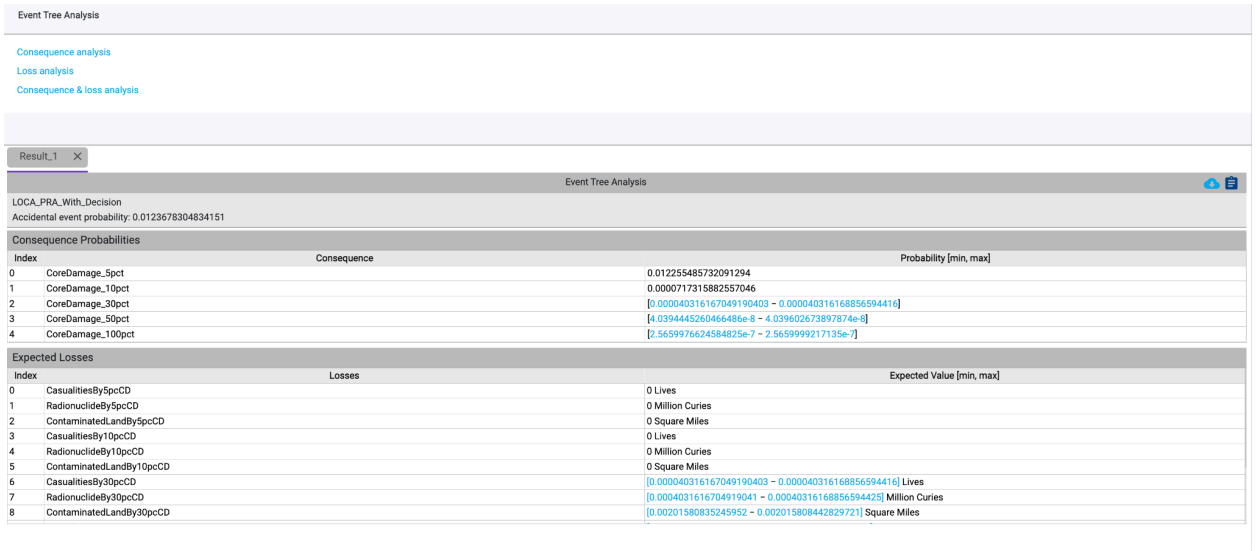
Select the property that you want to verify. E.g. click “Consequence analysis”, and the following popup will appear:



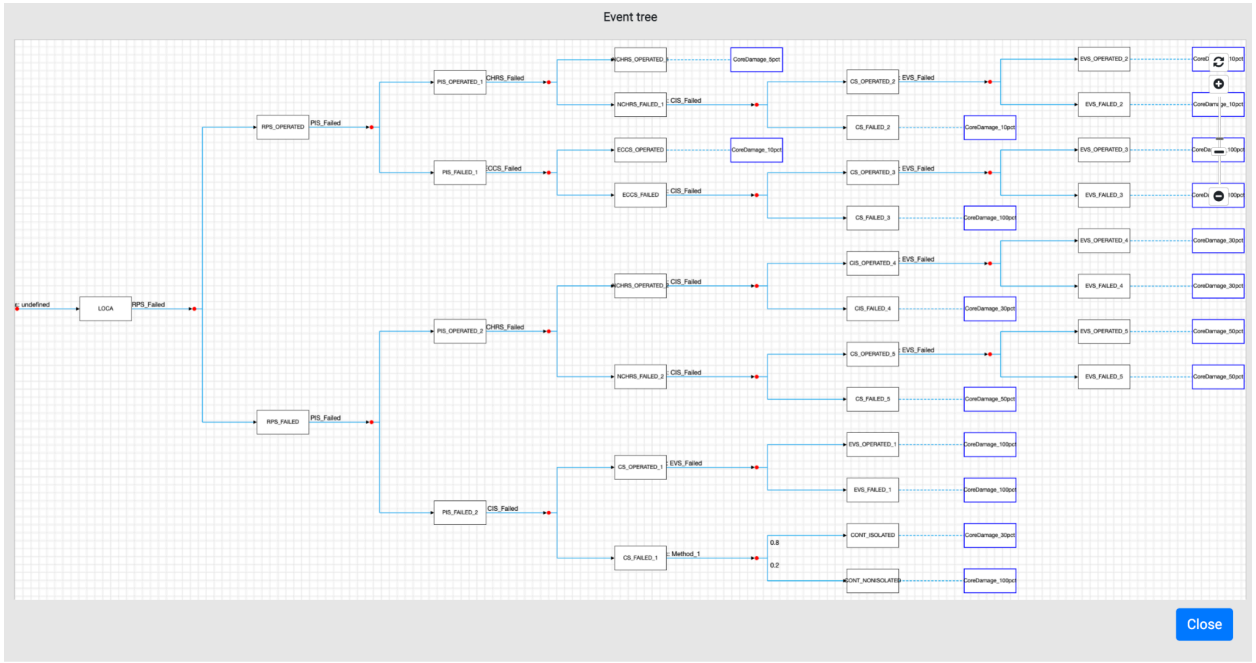
- Select the event tree model that you want to analyze from the "Event tree" dropdown – it shows all event tree models that exist under the "Even Trees" in the left panel.
- Change the parameter set if needed. The selected parameter set should contain all constants/expressions that have been used in the model as parameters.
- The constants defined in the selected parameter set (above) are shown in the table, which can be updated if needed.
- Change the FT events set if needed. The selected FT event set should contain all FT events/expressions that are you used to specify the branching probabilities in the event tree.

- The constants defined in the FT Event set (above) are shown in the table, which can be updated if needed.
- Enter the name of the tab where the results of the analysis will be displayed.

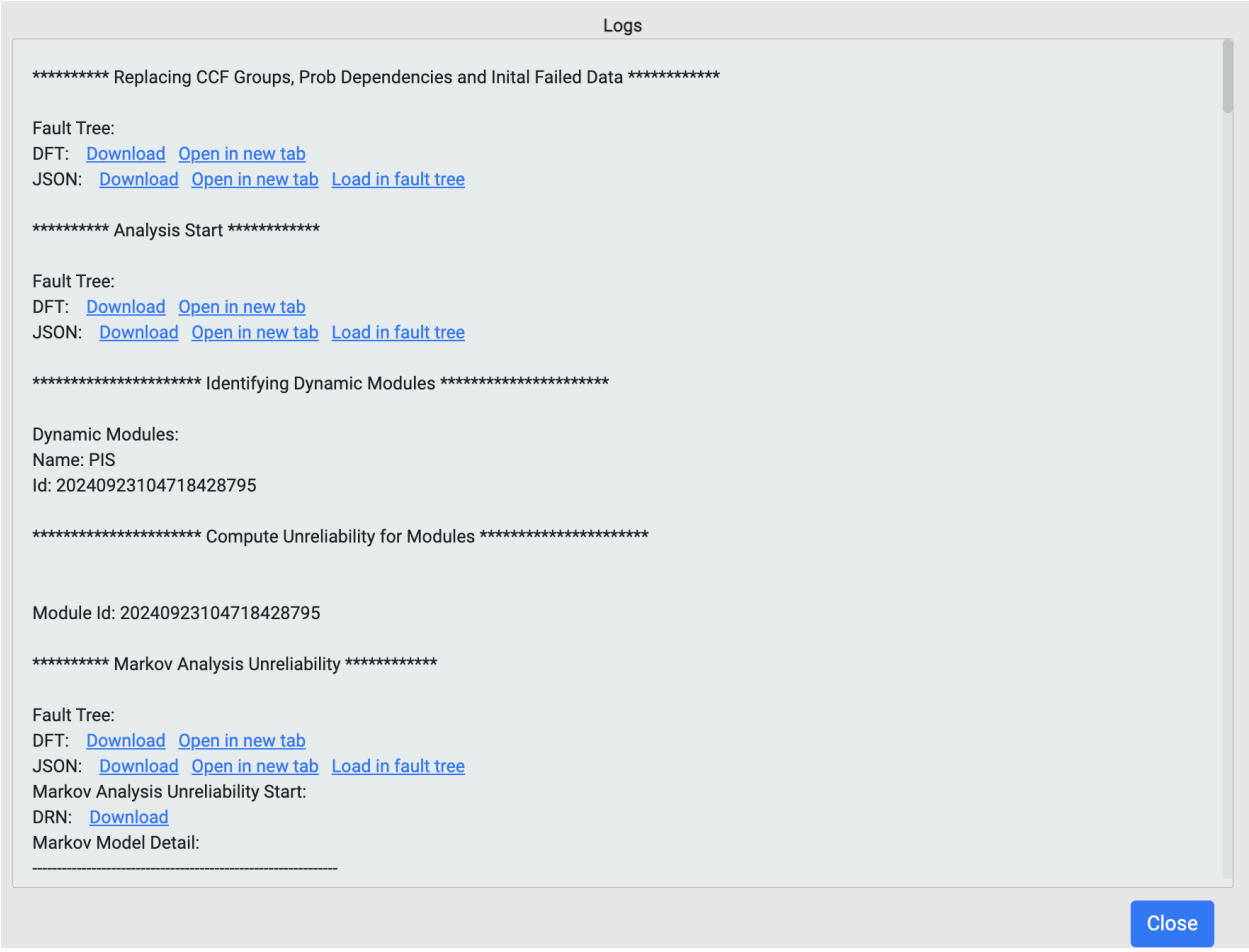
- Click the “Start” button to start the analysis.



- Data, which has maximum/minimum values, one can see which sequence of decisions at states will maximize/minimize the data. Click on the data link to see an event tree that maximizes/minimizes the value.



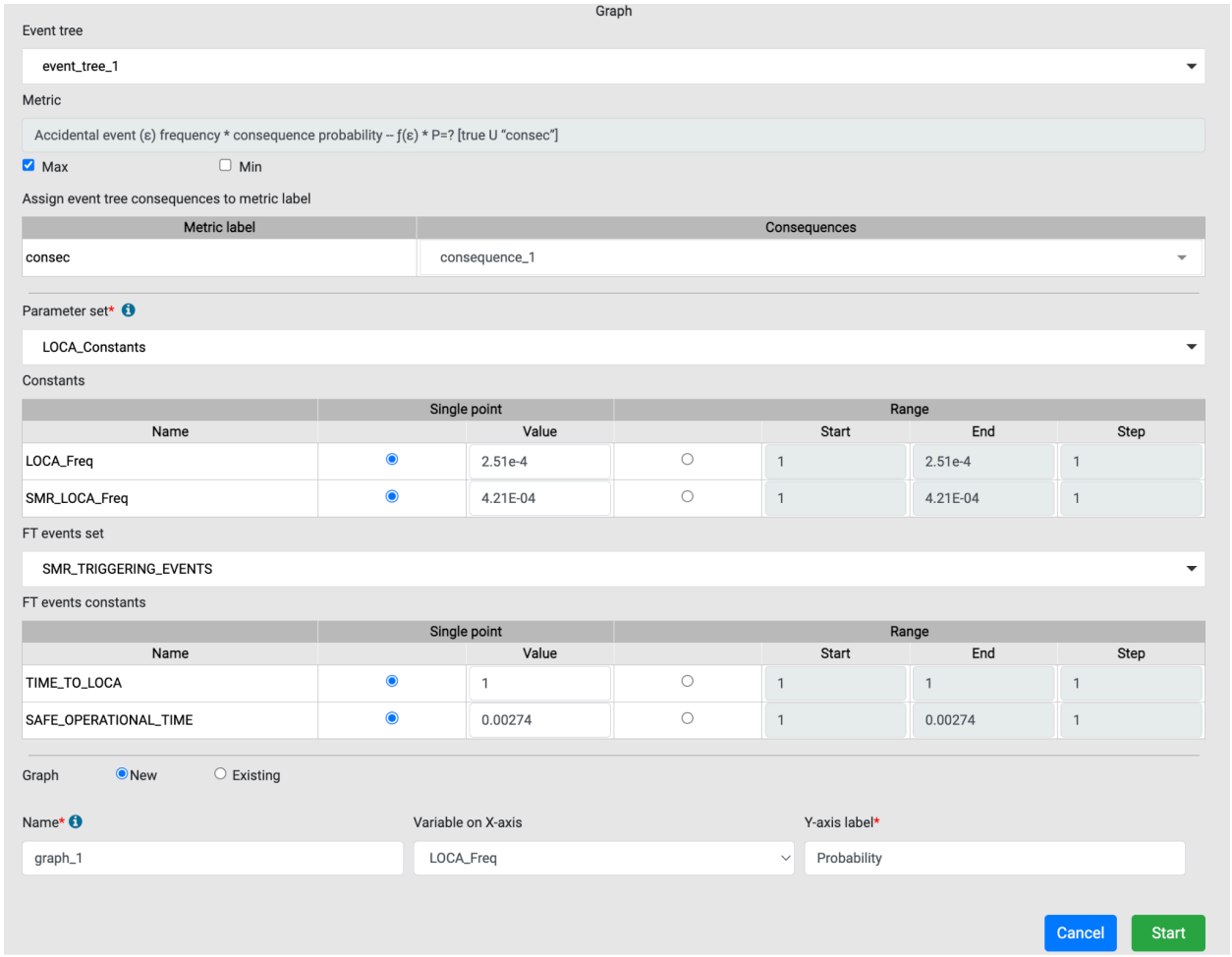
- Click the  icon to view the analysis log.
  - Click the “Download” link to download the generated artifact (DFT/DRN).
  - Click the “Open in new tab” link to open the generated artifact in a new browser tab.
  - Click the “Load in fault tree” link to load the generated fault tree in the current project.



- Click the icon to download the results in a CSV format.

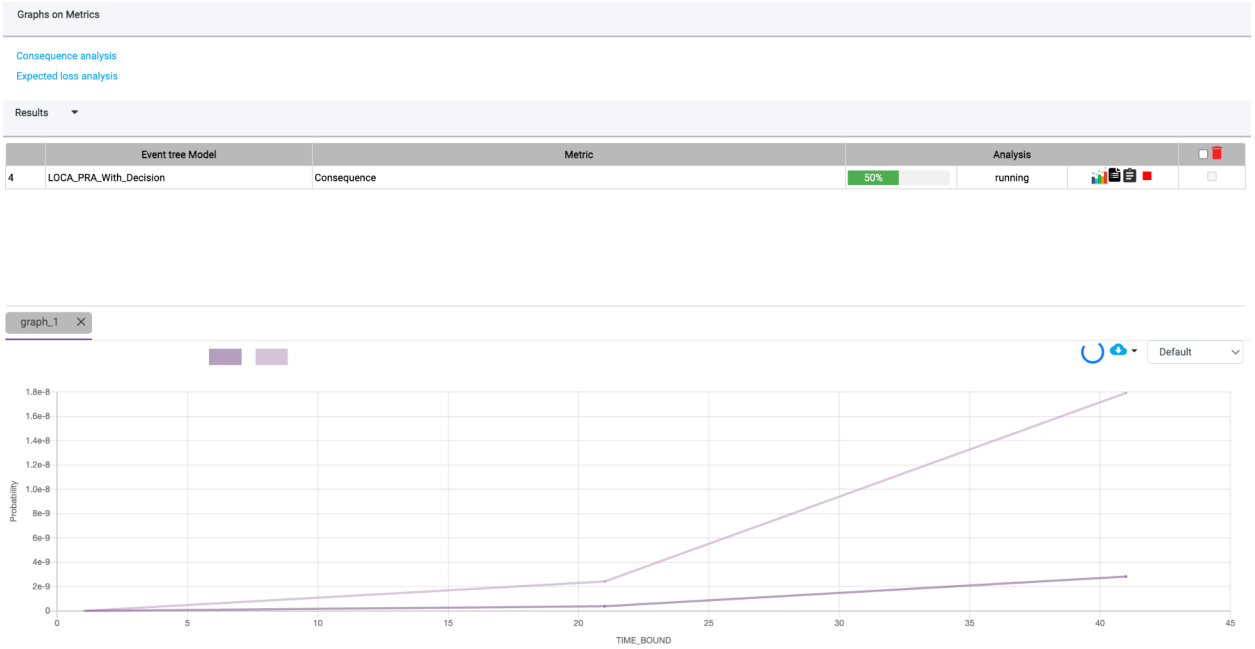
Graphs

We provide an interface to plot consequence probabilities/frequencies or expected values of loss quantities against different parameters of interest. Click on the “Graphs” link under “Computing” in the left panel. Click the “Consequence/Expected Loss”, and the following window will appear:

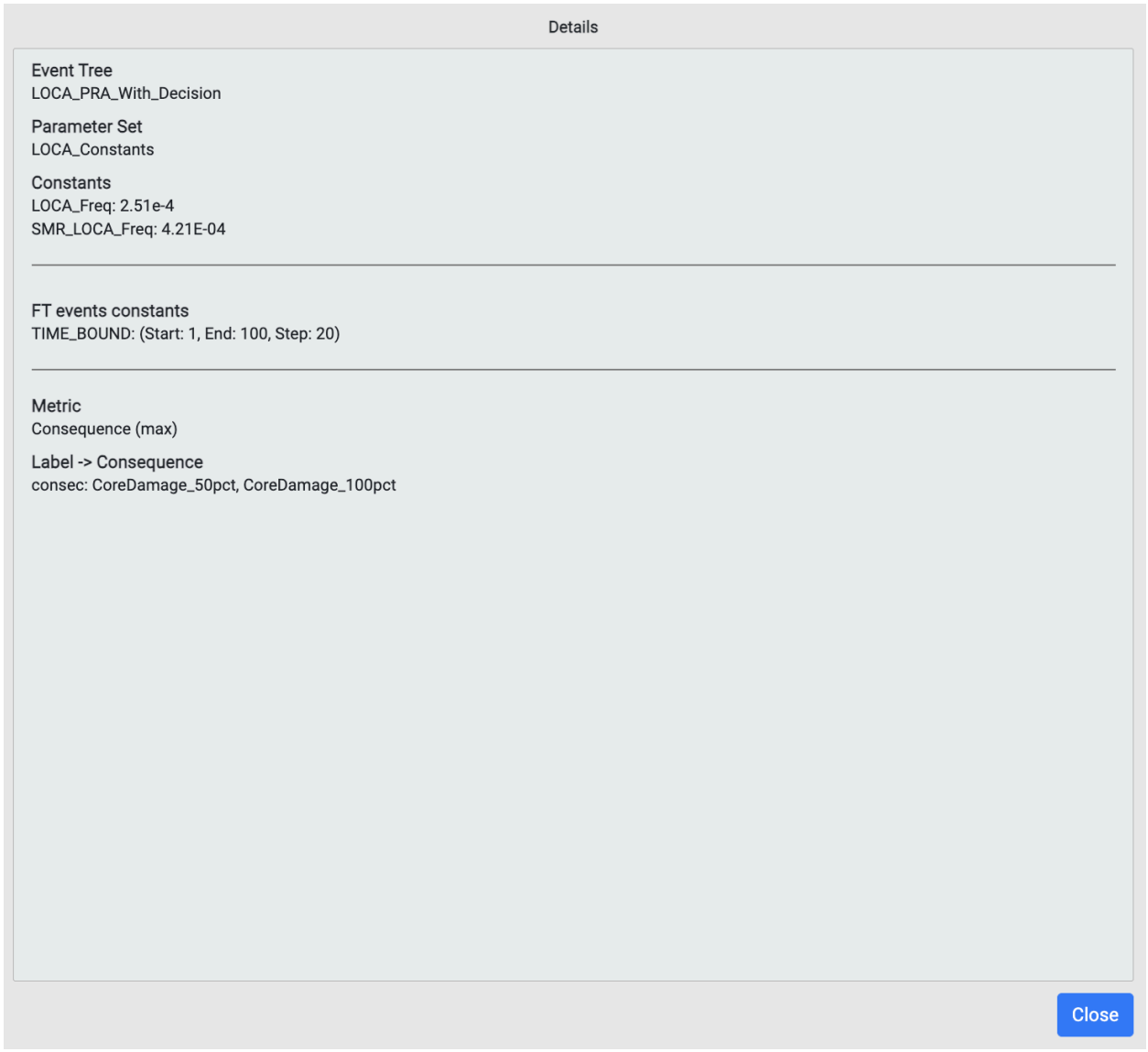


- "Event tree" dropdown: an event tree that is selected as a default in the "Event trees" page is automatically selected.
- Select whether the min, max, or both values of consequences probabilities/frequencies (expected loss) are to be computed.
- "Assign event tree consequences (quantities) to metric labels (parameter)": Select (multiple) consequence(s) (quantities) from the dropdown for which we want to measure probabilities/frequencies (expected values).
- A parameter set that is attached to the selected event tree is automatically selected. It can be changed at this point to generate another variant of the model.
- One can specify a range of values of constants defined in the selected parameter set.
- An FT event set which is attached to the selected event tree is automatically selected. It can be changed at this point to generate another variant of the model.
- One can specify a range of values of constants defined in the selected FT event set.
- Select whether to draw a new graph or it is to be plotted on an existing graph that has the same variable on X-axis.
- The variable on the X-axis of the graph can be either time-bound or from the constraints of the selected parameter/FT event set.

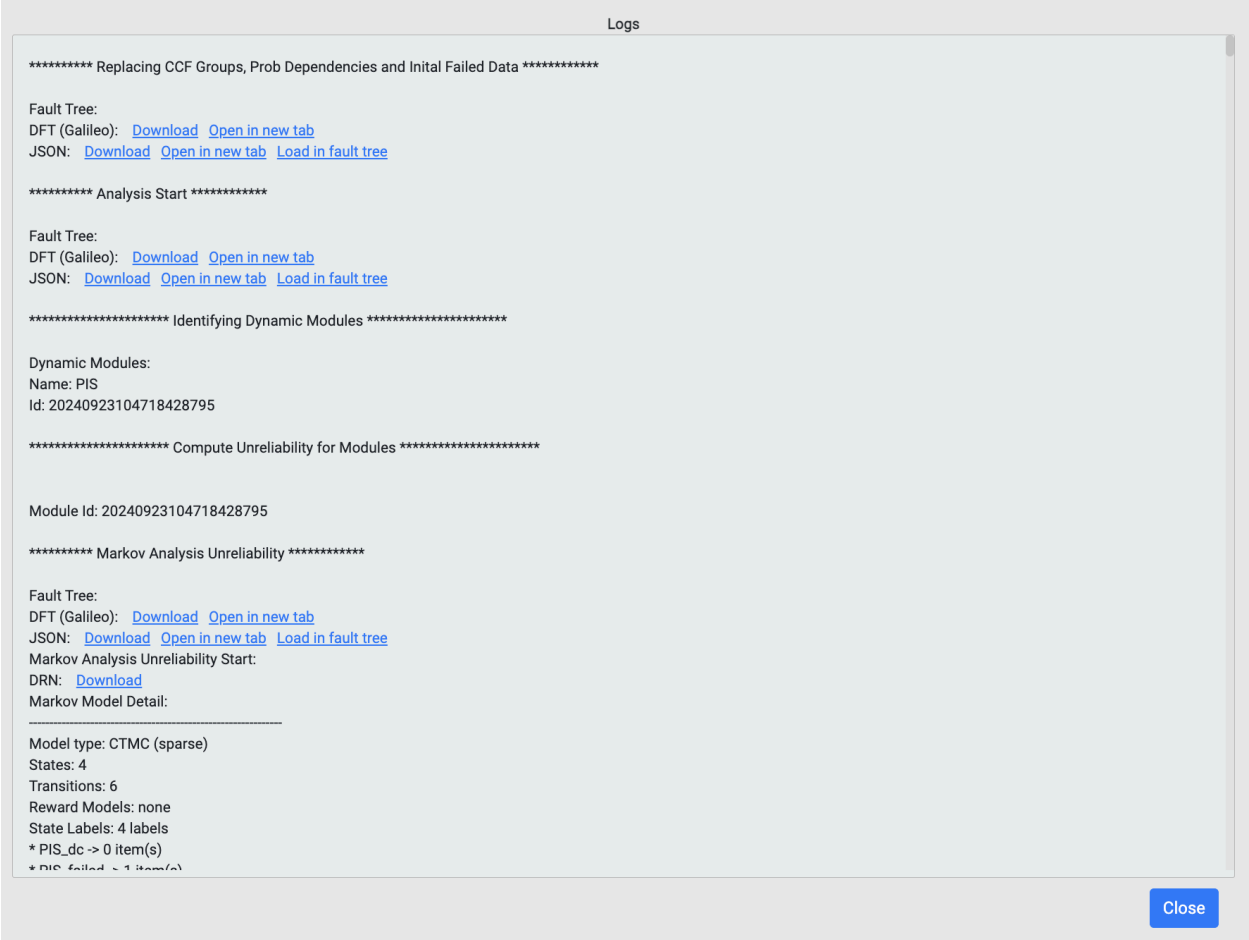
- Click the “Start” button to display the graph:



- Click the ■ icon to stop the running analysis.
- Click the ↻ icon to rerun the analysis and draw a graph.
- Click the 📄 icon to view the configuration of the analysis.



- Click the icon to view the analysis log.

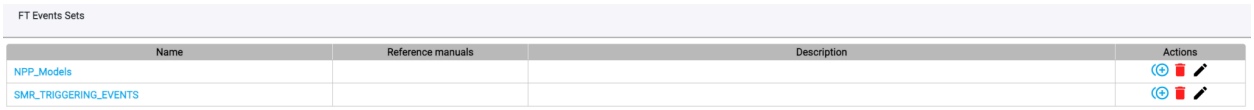


- Click the “Download” link to download the generated artifact (DFT/DRN).
- Click the “Open in new tab” link to open the generated artifact in a new browser tab.
- Click the “Load in fault tree” link to load the generated fault tree in the current project.

Interfaces

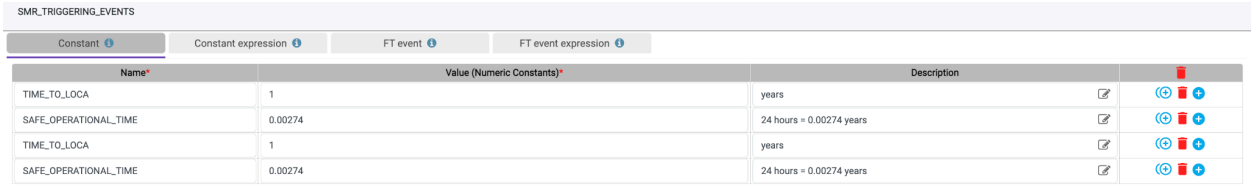
Interfaces can be defined to use fault trees within event trees. Labeled events (basic or compound) of fault trees can be associated with the branches of event trees as triggering events. The probabilities of these events provide the branching probabilities of the related transitions. To do this, we choose fault tree events and define CSL metrics to compute their probabilities. During the analysis phase, the metrics are calculated on the fault trees to provide the branching probabilities of the linked transitions.

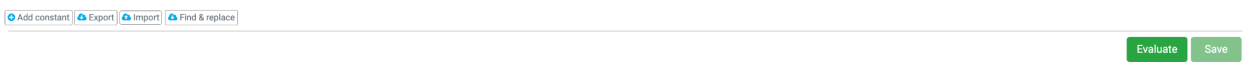
- Click on “FT Events sets” under “Interfaces” in the left panel.



- Click on the icon  to duplicate the FT event set.
- Click on the FT event set link to update it.
- Click “Add FT event set” to create a new event set. FT event set has:
    - Constants
    - Constant expressions
    - FT events, and
    - FT event expressions

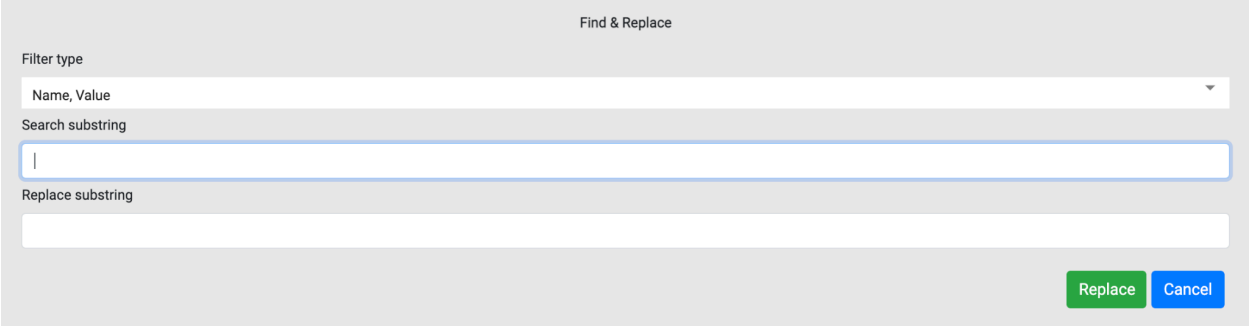
Constants



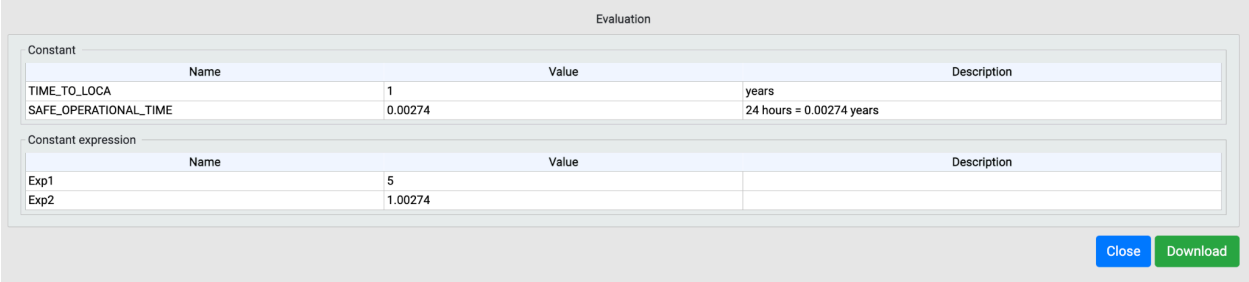


Constants can only be numeric e.g. 4, 2.3, 4e-6 etc. Their value can be changed at the time of analysis. For example, graphs can be plotted for matric results against ranges of values of constants.

- Click “Add constant” to enter a new row in the table.
- Click “Export” to export constants in a CSV format.
- Click “Import” to import constants from a CSV file.
- Click the “Find & replace” button to search sub-strings among Constants and replace them with a new string.

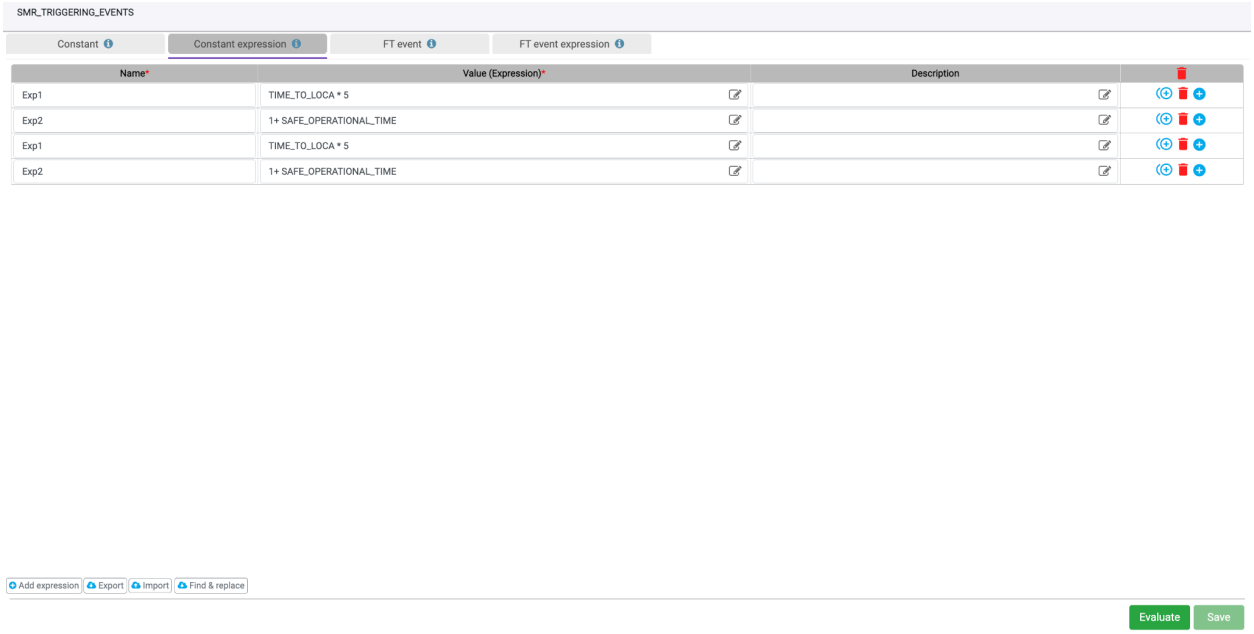


- Click the icon in the last column of any row to add a new row above it.
- Click the icon to duplicate the row.
- Click the “Evaluate” button to evaluate expressions in all tabs. The result will be displayed in a popup.



- Click the “Save” button to save the data. This action will save data in all the tabs.

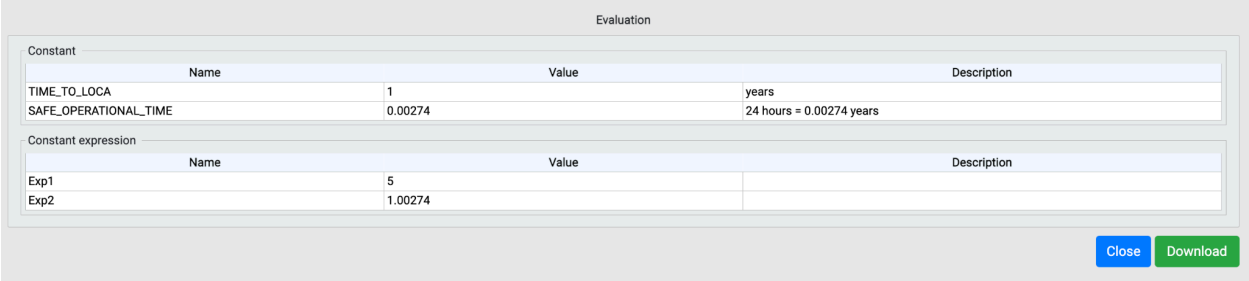
Constant Expressions



These are non-negative, real-value expressions, which can use constants (defined in Constants tab) e.g. x + 2 where x is a constant. The grammar of the expression is given here.

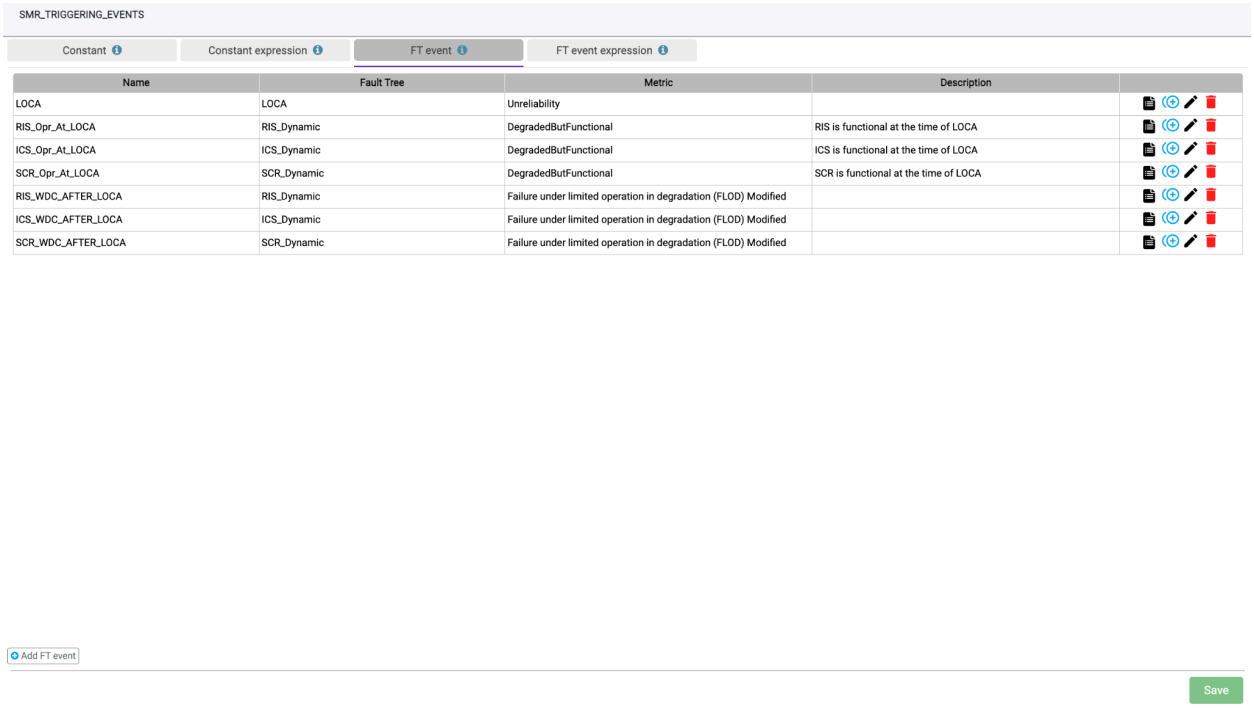
- Click the “Add expression” button to enter a new row in the table.
- Click the “Export” button to export expressions in a CSV format.
- Click the “Import” button to import expressions from a CSV file.
- Click the “Find & replace” button to search sub-strings among Constant Expressions and replace them with a new string.

- Click the  icon to duplicate the row.
- Click the  icon in the last column of any row to add a new row above it.
- Click “Evaluate” button to evaluate expressions in all tabs. The result will be displayed in a popup.

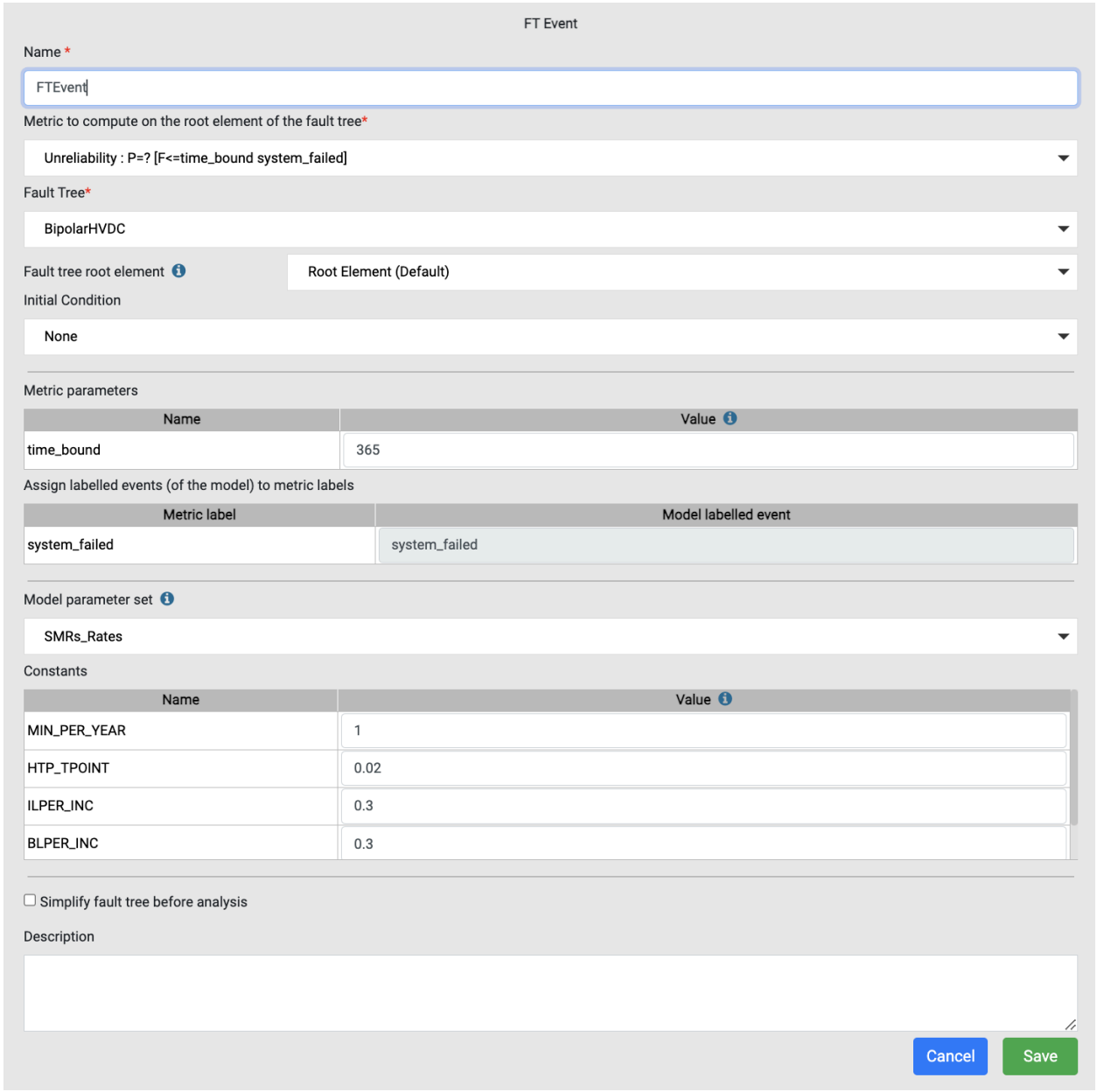


- Click the “Save” button to save the data. This action will save data in all the tabs.

FT event



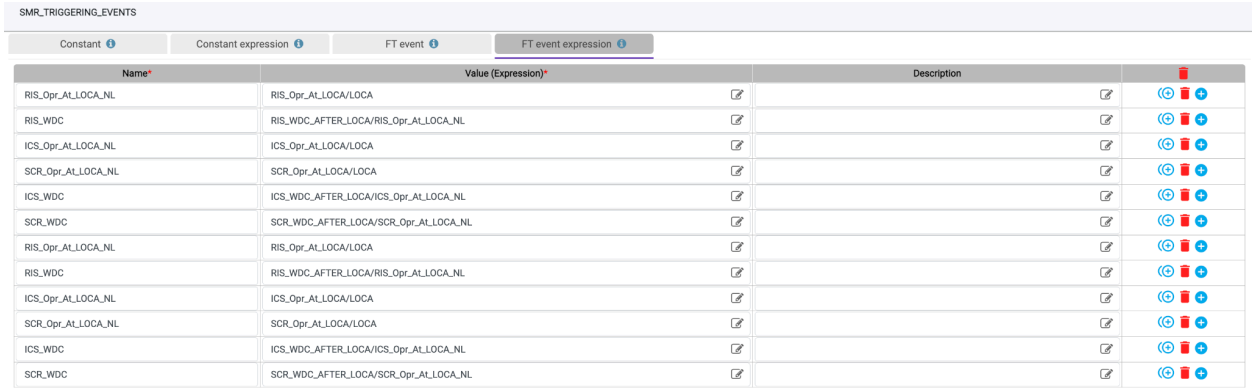
- Click “Add FT event” to create an FT event.



- Enter an event name that should be unique among other FT events.
- Select a metric to quantify the event
- Select a fault tree on which the metric is to be computed
- Select the root element of the fault tree
- Select an Initial condition for the fault tree
- Specify values of metric parameters. Note that Constants and Constant Expressions defined in the first two tabs can be used as values of metric parameters. These values can be changed at the time of event tree analysis.
- Select a parameter set for the fault tree model, and change the values of the constants defined in the parameter set if needed.
- Click the Save button to save the FT event.

- Click the icon  to view the configuration of the FT event.
- Click the icon  to duplicate an FT event.

FT event expression



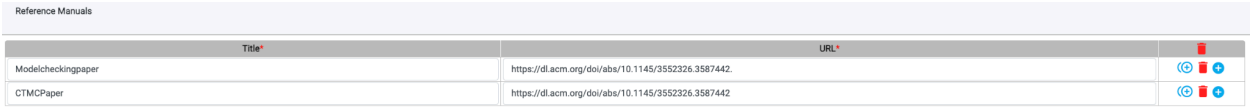
- Click the “Add FT event expression” to create a new row in the table.
- An expression can be defined using Constants, Constant expressions, and FT events defined in the previous tabs.
- Click the “Find & replace” button to search sub-strings among Constants and replace them with a new string.

- Click the icon  to duplicate an FT event expression.
- Click the icon  to create a new row above the current row.

Manuals

Link of reference manuals can be created and then associated with elements of fault trees/event trees.

- Click on the Manuals link in the left panel.
  - Click the “Add manual” button to create a new row in the table.
  - Click the “Import/Export” button to import/export data in the table.
  - Click the icon  to duplicate an FT event expression.
  - Click the icon  to create a new row above the current row.



Annotation of SysML Models with Safety Information

To annotate SysML model elements with safety information, we have created a few packages, which are to be used inside the SysML models against which fault trees are to be generated. These packages are:

- DGBMetadata: It contains a package DFTElements with the following sub-packages and elements:

- DFTGates package: It defines all gates that are used to construct fault trees.

```
package DFTGates {

    metadata def AND;
    metadata def OR;
    metadata def VOT {
        /* if any k-out-of-n components fail (input events),
        the system will fail (output event) n number of input events */
        attribute k : Number;
    }
    metadata def SPARE;
    metadata def PAND;
    metadata def POR;
    metadata def FDEP {
        /* The failure of the trigger_element renders the children of FDEP failed
        as per the value of probability attribute. The default value of probability
        is 1. */
        occurrence trigger_element;
        attribute probability: Real;
    }
    metadata def FSEQ;
    metadata def MUTEX;
}
```

- DFTBEs package: It defines all basic elements that may be used in fault trees.

```
package DFTBEs{
    abstract metadata def BE{
        /* The value of dormancy attribute is only relevant if the BE is a
        child of a spare spare gate. The default value of dormancy is 1. */
        attribute dormancy:Real;
    }
    abstract metadata def BE_CONSTANT_DISTRIBUTION specializes BE {
        attribute prob:Real;
    }
    abstract metadata def BE_EXPONENTIAL_DISTRIBUTION specializes BE {
        attribute rate:Real;
    }
    abstract metadata def BE_ERLANG_DISTRIBUTION specializes BE {
        attribute rate:Real;
        attribute phases:Real;
    }
    abstract metadata def BE_NORMAL_DISTRIBUTION specializes BE {
        attribute mean:Real;
        attribute stddev:Real;
    }
    abstract metadata def BE_WEIBULL_DISTRIBUTION specializes BE {
        attribute rate:Real;
        attribute shape:Real;
    }
}
```

    - TOP_LEVEL metadata: It is used to annotate an element of a fault tree as a top-level element. More than one element can be annotated as top-level elements. This helps generate multiple fault trees (for different scenarios) collectively that may share Gates and BEs.
- FailureModes: It defines all failure modes that may be used to annotate elements of SysML models with safety information. At the moment we allow failure modes to be modeled with the following failure distributions:

- Exponential distribution
- Erlang distribution
- Weibull distribution
- Log-normal distribution, and
- Constant distribution

Moreover, within this package, we allow to define model constants as (DFTParameters) enumerations. These constants can be used to define failure rates, probabilities, shapes, etc. of failure modes.

```
public import DGBMetadata::DFTElements::*;
/* DFTParameters enumiration defines constants used to annotat
failure rates/probabilites/shares/etc. of BEs in fault trees.
enum def DFTParameters :> Real {
    FIT1 = 0.000000001;
    FIT2 = 0.000000002;
    FIT3 = 0.000000003;
    FIT4 = 0.000000004;
    prob = 0.2;
    param1 = 10.2;
    param2 = 1.1;

}

metadata def FIT1 specializes BE_EXPONENTIAL_DISTRIBUTION{
        attribute redefines rate = DFTParameters::FIT1;
}
metadata def FIT2 specializes BE_EXPONENTIAL_DISTRIBUTION{
        attribute redefines rate = DFTParameters::FIT2;
}
metadata def FM1 specializes BE_CONSTANT_DISTRIBUTION{
        attribute redefines prob = DFTParameters::prob;
}
metadata def FIT3 specializes BE_ERLANG_DISTRIBUTION{
        attribute redefines rate = DFTParameters::FIT3;
        attribute redefines phases = 2;
}
metadata def FM_4 specializes BE_NORMAL_DISTRIBUTION{
        attribute redefines mean = DFTParameters::param1;
        attribute redefines stddev = DFTParameters::param2;
}
metadata def FIT4 specializes BE_WEIBULL_DISTRIBUTION{
        attribute redefines rate = DFTParameters::FIT4;
        attribute redefines shape = 3;
 }
```

Laptop Example.

The following example explains how elements within the SysML model can be annotated to generate fault trees out of them.

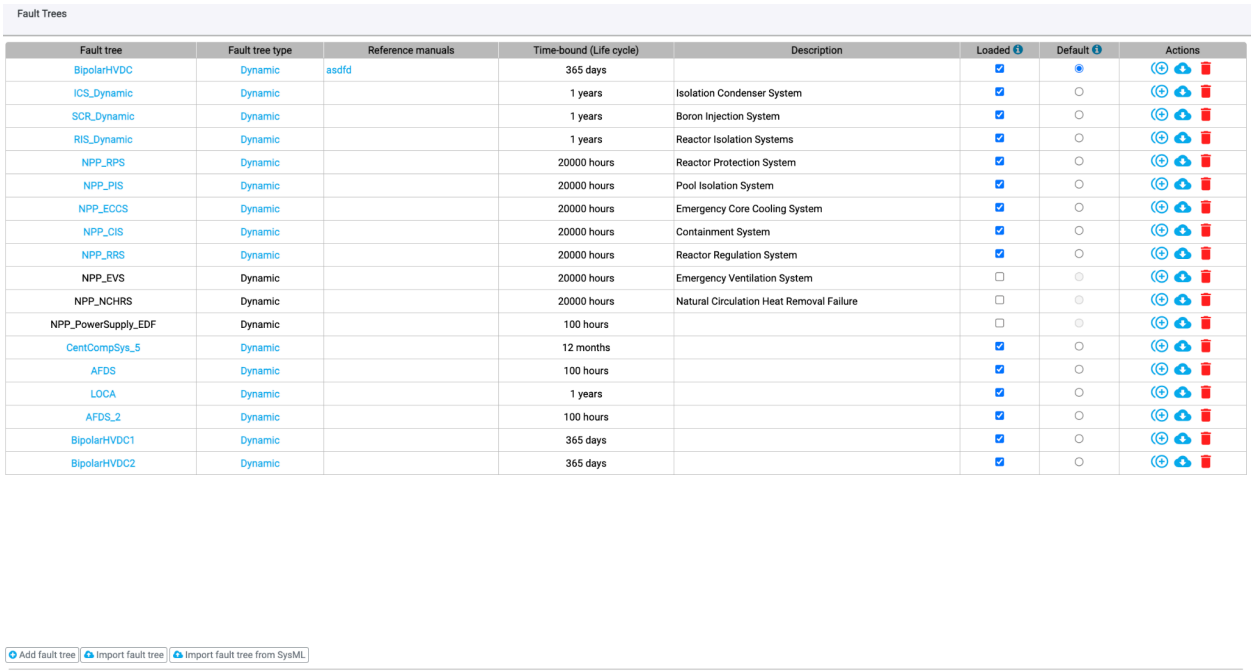
```
package LaptopPackage {
    private import FailureModes::*;
    part Laptop {
        part CPU1 {
            metadata Failure:FIT2;
        }
        part CPU2 {
            metadata Failure:FIT1;
        }
        part cooling {
            metadata Failure:FIT1;
        }
        part plug {
            metadata Failure:FIT2;
        }
        part battery {
            metadata Failure:FIT2;
        }
        part switch {
            metadata HWF:FIT1;
        }
        metadata power:SPARE about plug::Failure, battery::Failure;
        metadata processor:AND about CPU1::Failure, CPU2::Failure;
        metadata laptop:OR about power, processor;
        metadata Dep:FDEP about CPU1::Failure, CPU2::Failure {
            trigger_element = cooling::Failure;
        }
        metadata TLE1:TOP_LEVEL about laptop;
        metadata TLE2:TOP_LEVEL about power;
    }

}
```

After the compilation of the SysML model annotated with safety information using our packages in Jupyter Notebook, run the following command to export the package in JSON format. **Currently, we support the latest version – v0.45.0 – of SysML 2.0:** SysML-v2-Release

%export <package_name>

After downloading, it can be uploaded inside the SAFEST tool at the “Failure Models” page under “Fault Tree Analysis” in the left panel:


Fault Trees

Fault tree	Fault tree type	Reference manuals	Time-bound (Life cycle)	Description	Loaded	Default	Actions
BipolarHVDC	Dynamic	asdfd	365 days		☑	◉	
ICS_Dynamic	Dynamic		1 years	Isolation Condenser System	☑	○	
SCR_Dynamic	Dynamic		1 years	Boron Injection System	☑	○	
RIS_Dynamic	Dynamic		1 years	Reactor Isolation Systems	☑	○	
NPP_RPS	Dynamic		20000 hours	Reactor Protection System	☑	○	
NPP_PIS	Dynamic		20000 hours	Pool Isolation System	☑	○	
NPP_ECCS	Dynamic		20000 hours	Emergency Core Cooling System	☑	○	
NPP_CIS	Dynamic		20000 hours	Containment System	☑	○	
NPP_RRS	Dynamic		20000 hours	Reactor Regulation System	☑	○	
NPP_EVS	Dynamic		20000 hours	Emergency Ventilation System	☐		
NPP_NCHRS	Dynamic		20000 hours	Natural Circulation Heat Removal Failure	☐		
NPP_PowerSupply_EDF	Dynamic		100 hours		☐		
CentCompSys_5	Dynamic		12 months		☑	○	
AFDS	Dynamic		100 hours		☑	○	
LOCA	Dynamic		1 years		☑	○	
AFDS_2	Dynamic		100 hours		☑	○	
BipolarHVDC1	Dynamic		365 days		☑	○	
BipolarHVDC2	Dynamic		365 days		☑	○	

Add fault tree | Import fault tree | Import fault tree from SysML

Click the “Import failure model from SysML” button to extract failure models as well as parameter sets from the SysML model. The failure models are added to the list of existing failure models on the “Failure Models” page. Whereas all constants (DFTParameters enumerations inside the SysML FailureModes package) are added to the existing parameter sets on the “Parameter Sets” page.

Grammars

Regular Expressions of Identifiers and Numeric Constants

- Identifier (id):
  - It is used to give a unique name to e.g. constants, expressions, etc.
  - It is a string of characters starting with a capital letter (A-Z), a small letter(a-z), or an underscore (_) followed by a capital letter(s), a small letter(s), underscore or digit(s) (0-9).
  - Note that identifiers cannot be from the list of keywords of a grammar.
- Numeric constant (nc):
  - Simple,decimals and exponential i.e 123, 123.123, 123e+1, 123e-1, 123e1, 123.123e+1, 123.123e-1, 123.123e1, 123.123E1, 0.12

Context Free Grammar (CFG) of Expressions

- RE → E | + nc | - nc
- E → E OP E | (E ? E: E) | nc | id | (RE) | BF (RE, RE) | UF(RE) | ! E
- BF → pow, log, min, max
- UF → floor, ceil, abs
- OP → => | & | = | != | >= | <= | < | > | + | - | * | / | %
- OP → |

Where

- "id" is an identifier of an expression,
- "nc" is a numeric constant string, and
- pow, log, min, max, floor, ceil, and abs are grammar keywords.

Note: The above grammar can produce both boolean and real expressions; however, this tool only processes real expressions. In other words, while the grammar allows for expressions that may evaluate to either boolean (true/false) or real (numeric) values, the tool is specifically designed to handle only those expressions that result in real numbers. Boolean expressions, although included in the grammar, are not utilized by the tool.

Examples of real expressions:

- 2.2
- 13.12e-9
- 2 + x
- 3.9 * pow(x, y) + log (a, b)
- 4.5 / min(x*x, y+2) * max (a, b)
- 3 + floor(x) + ceil(y * a) + abs(b)
- (x > y ? 3 : a * b + c)

In these examples, each expression results in a real (numeric) value, aligning with the tool's requirements.

Context Free Grammar (CFG) of Boolean Expressions

- E → E OP E | id | (E) | !id | !(E)
- OP → |
- OP → &

Where

- "id" is an identifier of an expression, and

Context Free Grammar (CFG) of Continuous Stochastic Logic (CSL)

- PROP → P OP2 Type [PathFormula] | T OP2 Type [RewardFormula] | LongRun OP2 Type [StateFormula]
- Type → =? | OP3 E
- LongRun → LRA | S
- PathFormula → OP4 BoundedExpression StateFormula | StateFormula OP5 BoundedExpression StateFormula
- BoundedExpression → ^ { Bound } | { Bound } | Bound | NULL
- Bound → [E,E] | OP3 TIME
- TIME → (E) | nc
- RewardFormula → I = E | C <= E | F StateFormula | LongRun
- StateFormula -> StateFormula OP6 StateFormula | P OP2 OP3 E [PathFormula] | LongRun OP2 OP3 E [StateFormula] | id | (StateFormula) | true | !StateFormula
- OP → => | & | | | = | != | <= | >= | > | < | + | - | * | / | %
- OP1 → + | -
- OP2 → min | max | NULL
- OP3 → <= | >= | > | <
- OP4 → G | F
- OP5 → U | W | R
- OP6 → | | &
- E → E OP E | id | nc | (E) | !(E) | (OP1 nc)
- NULL → empty string

Where

- “id” is an identifier of an expression,
- “nc” is a numeric constant string, and
- true, false, Pmin, Pmax, Smin, Smax, Tmin, Tmax, LRAmin, LRAmax, P, R, T, S, LRA, min, max, G, U, F, W, C, I, failed, system_failed are keywords of the grammar.