

User Manual

(SAFEST Tool)

Table of Contents

[System failure models](#)

[System Modeling with SAFEST](#)

1. [Quantifiable States](#)
2. [Parameter Sets](#)
3. [Metrics](#)

[System analysis with SAFEST](#)

1. [Analysis](#)
2. [Bounded Analysis](#)
3. [Graphs](#)
4. [Interactive Simulation](#)
5. [Minimal Cut Sets](#)

System failure models

System failure models comprise many failure scenarios represented as fault trees. We find sub-scenarios for each scenario, which are represented by boolean equations on the components of fault trees.

A mathematical logic is used to define several metrics that are used to assess the probability of failure scenarios. For this, we employ continuous stochastic logic (CSL). The important metrics are e.g. (conditional) dependability, mean-time-to-failure, average failure probability per hour, etc.

We can utilize parameters to specify the failure distributions of the fundamental elements, creating parametric fault trees, which allow us to observe how the chance of failure scenarios varies as the failure probabilities of the fundamental elements in the fault trees. Changing the values of fundamental elements allows us to produce several fault tree versions.

System modeling with SAFEST

In the SAFEST tool, system failure models are built as a project.

A SAFEST project comprises of multiple:

1. Failure models (scenarios). Each failure model comprises of:
 - a. A hierarchical fault tree – static or dynamic

- b. Quantifiable states – in the form of boolean equations on elements of the fault tree.
2. Parameter Sets – to generate several variants of fault trees. Each parameter set comprises of:
- a. Constants
 - b. Real-value expressions
 - c. Probability distributions (Exponential, Erlang, Weibull, Log-normal)
 - d. Empirical probability distributions – failure distributions generated from data sets.
3. Metrics. There is a list of predefined metrics (classified into basic, complex and criticality metrics). For advanced users, it is possible to define custom metrics using continuous stochastic logic (CSL).

Steps to build system failure models:

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

New Project

Name*

Version

Author

Department

Description

Extract fault trees from a SysML 2.0 model

Cancel
Create

2. Users have the option to extract failure models from SysML 2.0 models. In this case, a SysML 2.0 model, in which different failure scenarios are annotated is selected. The algorithm will automatically generate fault trees of these scenarios and upload them to the project “Failure Models”.
3. Fill up mandatory fields, and click the “Create” button. The following page will appear, where users can make any changes if required.

Project Information

Name*	Version	Author	Department
Test_Project	1.0	John	Electrical Engineering
Description			
Save			

4. Click on the “Failure Models” in the left panel to display all failure models in a table. For a new project, one failure model is created automatically as a dynamic fault tree.

Failure Models  

Model name	Fault tree	Version	Author	Time bound (Life cycle)	Description	Default 	Actions
Test_Project_model	Dynamic	1.0	John	100		<input checked="" type="radio"/>	 
Static_Tree	Static	1.2	Kareem	100		<input type="radio"/>	 

5. A failure model which is worked upon the most is set as a default model by selecting the corresponding radio button.
6. Failure models can be uploaded by clicking on the  near the “Failure Models” title. For downloading individual models, click the download icon in the respective row.
7. Click on the plus icons  near the “Failure Models” title to add a new failure model.

Failure Model

Name*

Version

Author

Time bound (Life cycle)* i

Parameter set i

Description

Fault tree type Dynamic Static

Cancel Save

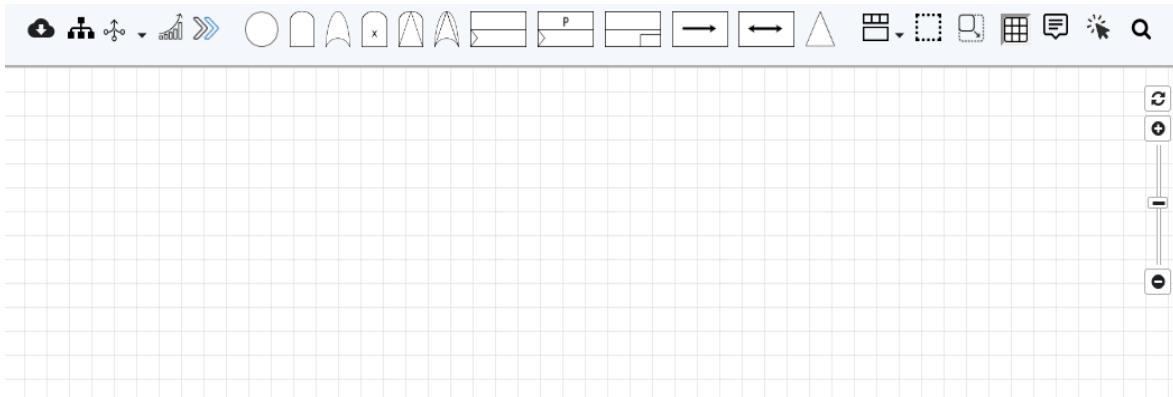
- A time bound for which model is to be analyzed may be inserted. This value can be changed at the time of analysis as well.
- A parameter set in which parameters to be used in the model are defined may be selected. It can also be changed at the time of analysis.
- Type of fault tree can be selected as “Static” or “Dynamic”.
- Click on the “Save” button to save a failure model.

8. Click on the failure model name in the left panel to see details about the model.

Model Information

Name*	Version	Author
Test_Project_model	1.0	John
Fault tree	<input type="radio"/> Static <input checked="" type="radio"/> Dynamic	
Time bound (Life cycle)* <small>i</small>	100	Parameter set <small>i</small>
Description	<input type="text"/>	
Save		

- Failure model fields can be changed on this screen and saved.
9. Click on the “Fault Tree” of the failure model to open a drag-and-drop grid.



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

Block

Name* ⓘ

Create a fault tree Import a fault tree from a file

Description

Cancel Save

MUTEX

Type

MUTEX

Name* ⓘ

Description

Cancel Save

SEQ

Type

SEQ

Name* ⓘ

I10

Description

Cancel Save

OR

Type

OR

Name* ⓘ

I3

OR gate propagates failure if any of its children will fail.

Description

Top level element Failure probability is quantifiable

Cancel Save

VOT

Type

VOT

Name* ⓘ

I4

Threshold(θ) * ⓘ

1

VOT gate propagates failure only if number of children that fail is greater than the threshold value.

Description

Top level element Failure probability is quantifiable

Cancel Save

FDEP

Type

FDEP

Name* ⓘ

I7

Description

Cancel Save

BE

Type

BE

Name*

I1

BE models the failure of a system component that cannot be decomposed further.

Enter failure distribution 

Select failure distribution 

Exponential distribution

Rate (λ) *

1

Enter dormancy (ζ) when used as a spare component*

1

Description

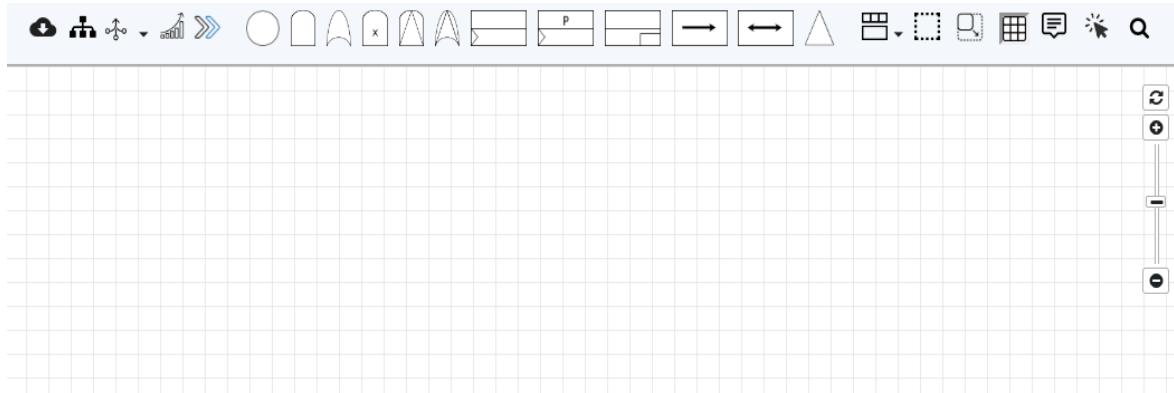
Top level element

Failure probability is quantifiable

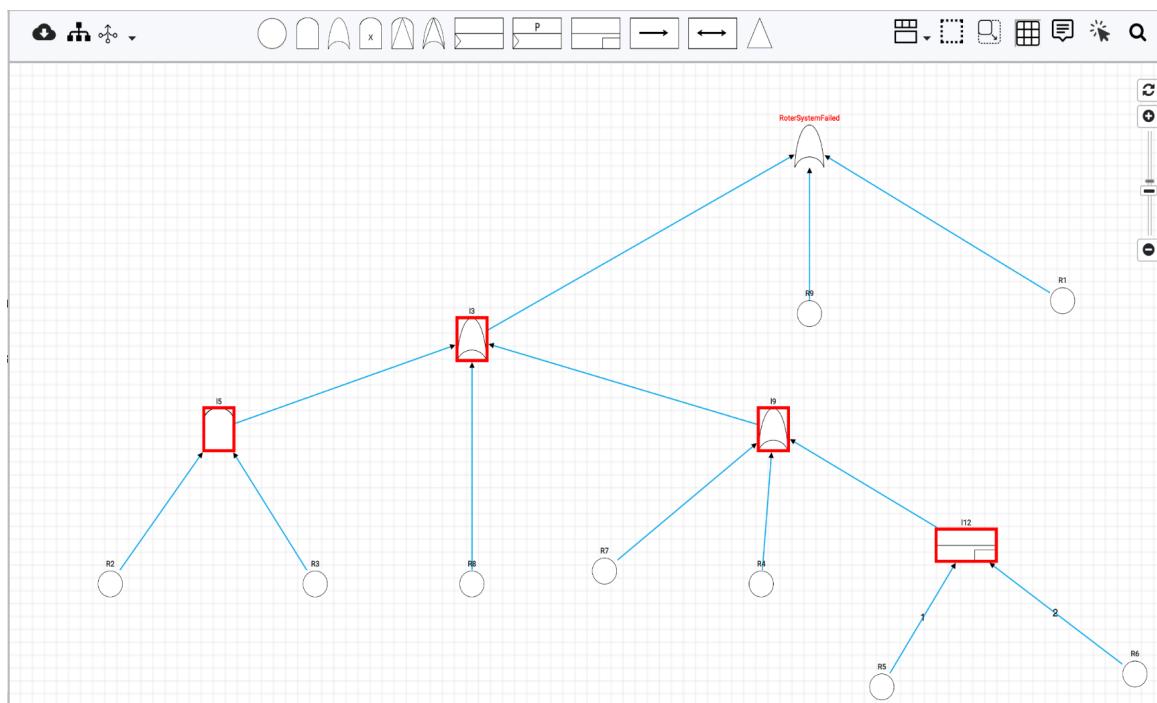
Cancel

Save

- Only one element can be selected as a top level element in a tree.
 - “Failure probability is quantifiable” checkbox is only visible in advance view.
 - Elements can be connected with each other by drawing edges between them.
 - Those elements which propagate failure to their parents can be selected (by selecting their “Failure probability is quantifiable” checkbox) to create sub-scenarios. This feature is only available in “Advance View”.



10. Click on the icon  (not visible when sub-trees are in focus) to highlight the elements that, along with their children, can be converted into modules (independent sub-trees). In order to convert an element and its children into a module, right click on it and then click "Make block".



11. In order to simplify a fault tree click on down arrow along the icon . It will give three options for tree simplification:

- Simplify by all rules. It will apply all rules recursively for simplification.
- Default rules. It will apply a selected set of rules (most commonly used) for simplification.
- Custom rules. It allows users to select rules that are to be applied for simplification.

Simplification Rules

<input type="checkbox"/>	Name	Description
<input type="checkbox"/>	SPLIT_FDEPS	Split FDEPs with two or more children into single FDEPs with only one child.
<input type="checkbox"/>	MERGE_BES	Try to merge BEs under an OR-gate into one BE.
<input type="checkbox"/>	TRIM	Trim parts of the DFT in place which do not contribute to the top level element.
<input type="checkbox"/>	REMOVE_DEPENDENCIES_TLE	Try to remove superfluous dependencies. These dependencies have a trigger which already leads to failure of the top level element.
<input type="checkbox"/>	MERGE_IDENTICAL_GATES	Try to merge gates with the same type and identical successors. These gates surely fail simultaneously and thus, one gate can be removed.
<input type="checkbox"/>	REMOVE_SINGLE_SUCCESSOR	Remove gates with just one successor. These gates will fail together with this child, so they can directly be eliminated.
<input type="checkbox"/>	FLATTEN_GATE	Flattening of AND-/OR-/PAND-gates.
<input type="checkbox"/>	SUBSUME_GATE	Subsumption of OR-gate by AND-gate or of AND-gate by OR-gate.
<input type="checkbox"/>	REPLACE_FDEP_BY_OR	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.
<input type="checkbox"/>	REMOVE_SUPERFLUOUS_FDEP	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.
<input type="checkbox"/>	REMOVE_SUPERFLUOUS_FDE...	Eliminate FDEP between two successors of an OR or PAND. Only supports FDEPs with one common predecessor.

Cancel
Simplify

12. Click on the icon (not visible when sub-trees are in focus) to do 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.

Analysis

Metrics *

Mean time to failure (MTTF): $T=? [F \text{ topLevelEvent}]$, Reliability: $1 - P=? [F \leq \text{time_bound} \text{ topLev...}]$

Failure model*

OWT

Metric parameters

Name	Value 
time_bound	100

Assign quantifiable-state labels (of the model) to metric labels

Metric label	Quantifiable-state label
topLevelEvent	failed (Root Element)

Model parameter set 

parameter_set1

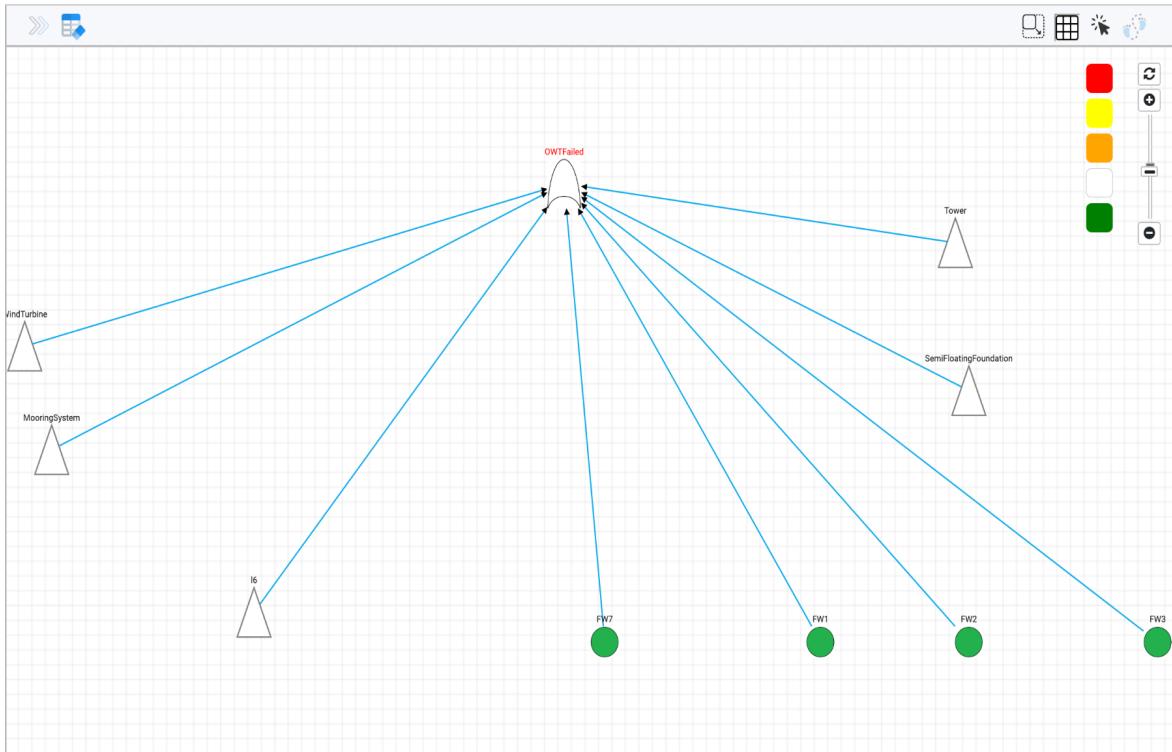
Analysis type: Markov BDD

Output tab: Existing New

Results

Cancel **Run**

13. Click on the icon  to start interactive simulation. It will take the user to the "Interactive Simulation" window under "Computing" in the left panel.



14. Click on the down arrow new the icon , it will give three 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

Tabular View

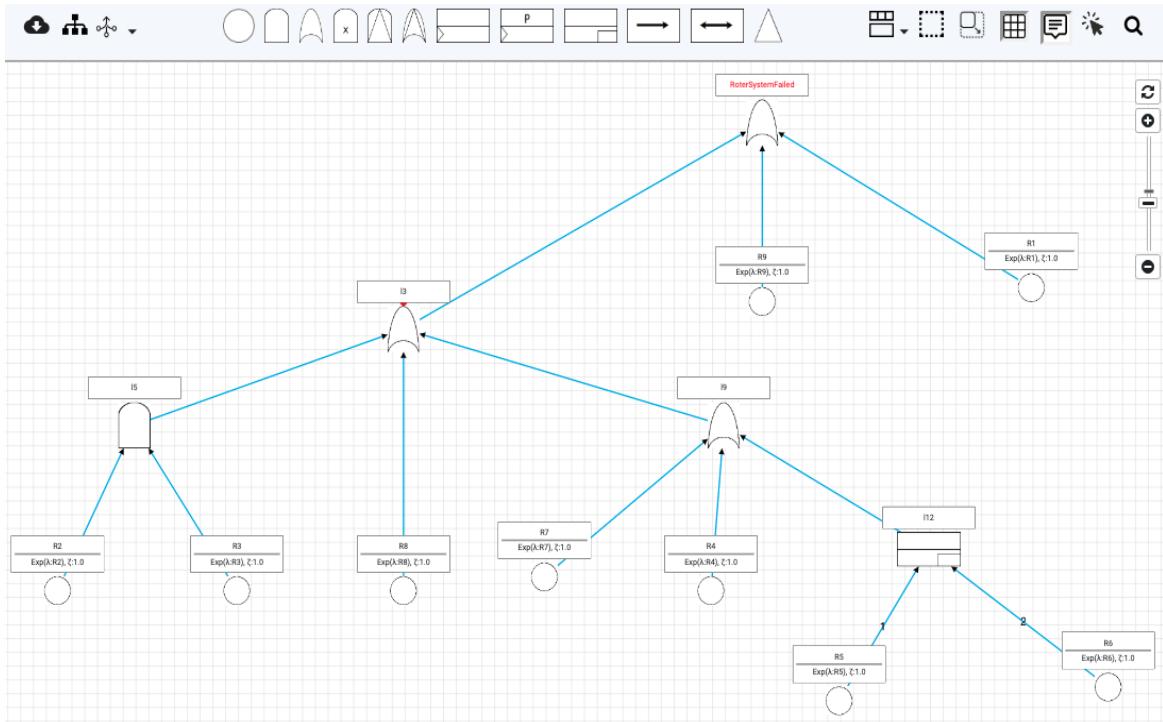
Fault Tree Elements			
BE: 4 Static Gates: 1 Dynamic Gates: 0 Blocks: 5			
Name	Type	Information	Description
OWTFailed	OR	toplevel: true	Floating offshore wind turbine system failure
FW7	BE	$\text{Exp}(\lambda: \text{FW7}, \zeta: 1.0, \text{quantifiable: true})$	Resonance
FW1	BE	$\text{Exp}(\lambda: \text{FW1}, \zeta: 1.0, \text{quantifiable: true})$	Transport
FW2	BE	$\text{Exp}(\lambda: \text{FW2}, \zeta: 1.0, \text{quantifiable: true})$	Fire
FW3	BE	$\text{Exp}(\lambda: \text{FW3}, \zeta: 1.0, \text{quantifiable: true})$	Ice rainfall
WindTurbine	BLOCK		
MooringSystem	BLOCK		
Tower	BLOCK		
SemiFloatingFoundation	BLOCK		
I6	BLOCK		Collision with other structures

c. Textual view. It shows the tree in Galileo format.



```
param G7;
param M5;
param R6;
param G1;
toplevel "OWTFailed";
"OWTFailed" or "I6_I6" "FW7" "FW1" "FW2" "FW3" "WindTurbine_WindTurbineFailed" "MooringSystem_MooringSystemFailed" "Tower_TowerFailed" "SemiFloatingFoundation_SFFailed";
"I6_I6" or "I6_FW4" "I6_FW5" "I6_FW6";
"FW7" lambda=FW7 dorm=1.0;
"FW1" lambda=FW1 dorm=1.0;
"FW2" lambda=FW2 dorm=1.0;
"FW3" lambda=FW3 dorm=1.0;
"WindTurbine_WindTurbineFailed" or "WindTurbine_W2" "WindTurbine_W3" "WindTurbine_W4" "WindTurbine_W7" "WindTurbine_W8" "WindTurbine_W9" "WindTurbine_W10" "WindTurbine_W11"
"WindTurbine_W12" "WindTurbine_RoterSystem_RoterSystemFailed" "WindTurbine_Sensors_SensorsFailed" "WindTurbine_Generator_GeneratorFailed";
"MooringSystem_MooringSystemFailed" or "MooringSystem_M7" "MooringSystem_I3" "MooringSystem_I4";
"Tower_TowerFailed" or "Tower_I2" "Tower_T1" "Tower_T2" "Tower_T3" "Tower_T10" "Tower_T4" "Tower_T9";
"SemiFloatingFoundation_SFFailed" or "SemiFloatingFoundation_F1" "SemiFloatingFoundation_I3" "SemiFloatingFoundation_I4" "SemiFloatingFoundation_F9";
"I6_FW4" lambda=FW4 dorm=1.0;
"I6_FW5" lambda=FW5 dorm=1.0;
"I6_FW6" lambda=FW6 dorm=1.0;
"WindTurbine_W2" lambda=8e-06 dorm=1.0;
"WindTurbine_W3" lambda=1e-05 dorm=1.0;
"WindTurbine_W4" lambda=1e-05 dorm=1.0;
"WindTurbine_W7" lambda=1.6e-05 dorm=1.0;
"WindTurbine_W8" lambda=2e-06 dorm=1.0;
"WindTurbine_W9" lambda=3.9e-05 dorm=1.0;
"WindTurbine_W10" lambda=1.3e-05 dorm=1.0;
"WindTurbine_W11" lambda=4e-06 dorm=1.0;
"WindTurbine_W12" lambda=3.3e-05 dorm=1.0;
"WindTurbine_RoterSystem_RoterSystemFailed" or "WindTurbine_RoterSystem_R9" "WindTurbine_RoterSystem_I3" "WindTurbine_RoterSystem_R1";
"WindTurbine_Sensors_SensorsFailed" vot2 "WindTurbine_Sensors_W61" "WindTurbine_Sensors_W62" "WindTurbine_Sensors_W63";
```

15. Click on the icon  to enable 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.
16. Click on the icon  to enable navigator at the bottom of the screen.
17. Click on the icon  to show the grid on the screen.
18. Click on the icon  to show the summary information about each element on the screen.



19. Click on the icon to display summary information about an element on hovering.
20. Click on the icon to search any element on the screen.

Quantifiable States

1. In advance view, “Quantifiable States” are visible in the left panel under each failure model. Click on the “Quantifiable States”, it will show a screen with two tabs:
 Basic: It will display all elements of the fault tree whose “Failure probability is quantifiable” checkbox is selected. Each element represents a sub-scenario about which metrics can be verified.

Quantifiable States			
Basic	Composite		
State Label	Element that fail in state	Description	
failed	Root Element		
FW7_failed	FW7		
FW1_failed	FW1		
WindTurbine_RoterSystem_RoterSystemFailed_mFailed_failed	WindTurbine_RoterSystem_RoterSystemFailed		

Composite: On this tab, we can create new sub-scenarios by writing boolean expressions on the basic scenarios shown in the Basic tab.

Quantifiable State

State label* ⓘ

Boolean expression * ⓘ

```
FW7_failed & FW1_failed &
WindTurbine_RoterSystem_RoterSystemFailed_failed
```

The above expression can use the following state labels.

newScenario

failed FW7_failed FW1_failed

WindTurbine_RoterSystem_RoterSystemFailed_failed

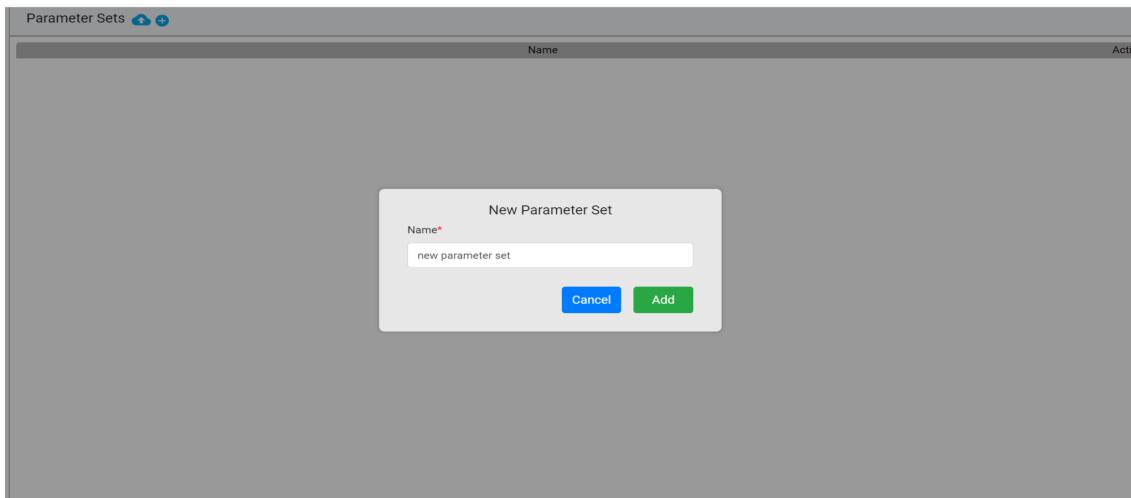
Description

Cancel **Save**

Quantifiable States		
Basic	Composite	
State Label	Boolean expression that characterize the state	Description
newScenario	FW7_failed & FW1_failed & WindTurbine_RoterSystem_RoterSystemFailed_failed	 

Parameter Sets

1. A parameter set contains constants, real-value expressions, failure distributions and empirical failure distributions (calculated from data sets), which are used inside fault trees. By changing the values of parameters, different variants of fault models can be created. Add a new parameter set by clicking on the plus icon.



After adding, you see a screen with tabs: constants, parameters (real-value expressions), failure distributions and empirical failure distributions.

- **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 matrix results against ranges of values of constants.

- **Real-value Expressions**

These are non-negative, real-value expressions, which can use constants (defined above) e.g. $x + 2$ where x is a constant. The grammar of expression is given [here](#).

- **Failure distributions**

Failure distributions can be exponential, erlang, Weibull, log normal, and constant probability. Multiple distributions can be added by pressing add row.

new parameter set

Failure distributions		Empirical failure dist.	
Constants	Parameters	Distribution	Description
distribution1	Exponential distribution	Rate (λ) 5	
distribution2	Erlang distribution	Rate (λ) $x+6$	Phases (k) 7
<input type="button" value="Add row"/> <input type="button" value="Export"/> <input type="button" value="Import"/>			
<input type="button" value="Save"/>			

- **Empirical failure dist.**

Empirical distributions are calculated from a data set using statistical methods. For example, a 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.

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

new parameter set

Failure distributions		Empirical failure dist.																	
Constants	Parameters	Distribution	Description																
<input type="button" value="Manage distributions"/>		<table border="1"> <thead> <tr> <th colspan="4">Computed Distributions</th> </tr> <tr> <th>Name</th> <th>Status</th> <th>Data</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td><input type="text" value="computedDistribution"/></td> <td><input type="button" value="Choose File"/> file.txt</td> <td><input type="button" value="Compute"/></td> <td><input type="button" value="Cancel"/></td> </tr> <tr> <td colspan="4"><input type="button" value="Sample Data"/></td> </tr> </tbody> </table>		Computed Distributions				Name	Status	Data	Action	<input type="text" value="computedDistribution"/>	<input type="button" value="Choose File"/> file.txt	<input type="button" value="Compute"/>	<input type="button" value="Cancel"/>	<input type="button" value="Sample Data"/>			
Computed Distributions																			
Name	Status	Data	Action																
<input type="text" value="computedDistribution"/>	<input type="button" value="Choose File"/> file.txt	<input type="button" value="Compute"/>	<input type="button" value="Cancel"/>																
<input type="button" value="Sample Data"/>																			
<input type="button" value="Add new"/>																			
<input type="button" value="Save"/>																			

After computations are done, update the data on the popup after pressing the refresh button. The newly computed distribution will become visible.

new parameter set

Failure distributions		Empirical failure dist.																	
Constants	Parameters	Distribution	Description																
<input type="button" value="Manage distributions"/>		<table border="1"> <thead> <tr> <th colspan="4">Computed Distributions</th> </tr> <tr> <th>Name</th> <th>Status</th> <th>Data</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>computedDistribution</td> <td>completed</td> <td>44.88 % Weibull (λ: 56.010013, η: 1.673703), 23.7 % Erlang (λ: 26.379431, k: 1.914421), 2.47 % LogN (μ: 3.638642, σ: 0.918553), 1.06 % Exp (λ: 50.50133)</td> <td><input type="button" value="Edit"/> <input type="button" value="Delete"/></td> </tr> <tr> <td colspan="4"> <input type="button" value="Add new"/> <input type="button" value="Refresh"/> <input type="button" value="Close"/> </td> </tr> </tbody> </table>		Computed Distributions				Name	Status	Data	Action	computedDistribution	completed	44.88 % Weibull (λ : 56.010013, η : 1.673703), 23.7 % Erlang (λ : 26.379431, k : 1.914421), 2.47 % LogN (μ : 3.638642, σ : 0.918553), 1.06 % Exp (λ : 50.50133)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>	<input type="button" value="Add new"/> <input type="button" value="Refresh"/> <input type="button" value="Close"/>			
Computed Distributions																			
Name	Status	Data	Action																
computedDistribution	completed	44.88 % Weibull (λ : 56.010013, η : 1.673703), 23.7 % Erlang (λ : 26.379431, k : 1.914421), 2.47 % LogN (μ : 3.638642, σ : 0.918553), 1.06 % Exp (λ : 50.50133)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>																
<input type="button" value="Add new"/> <input type="button" value="Refresh"/> <input type="button" value="Close"/>																			
<input type="button" value="Add new"/>																			
<input type="button" value="Save"/>																			

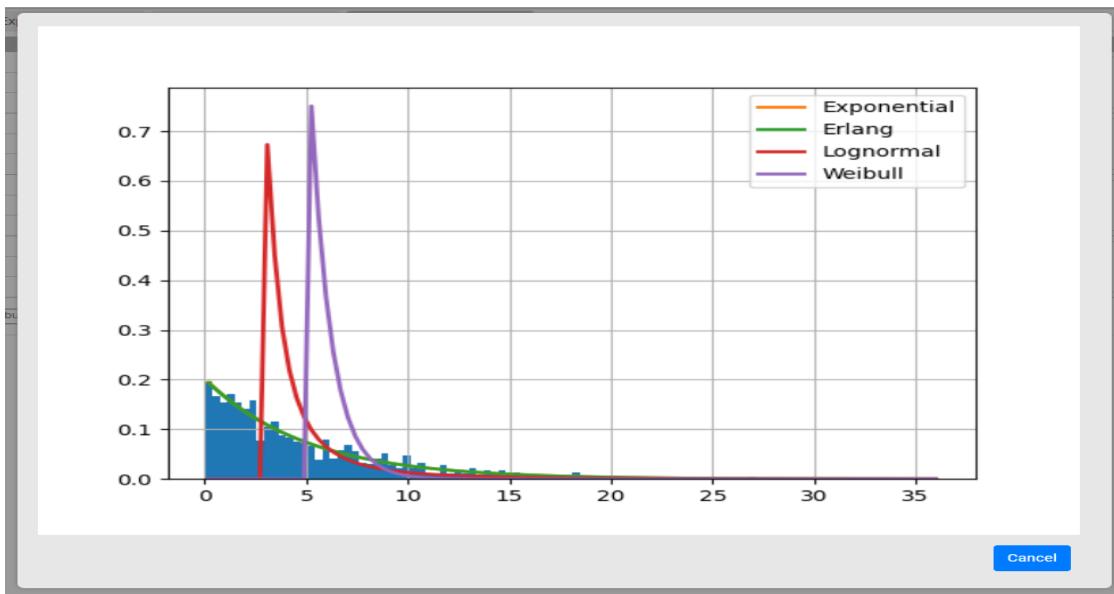
You can add these results as a new distribution in empirical distribution by pressing  icon.

Constants <small>i</small>		Parameters <small>i</small>		Failure distributions <small>i</small>		Empirical failure dist. <small>i</small>	
Name	Goodness to fit	Distribution	Description	Action			
computedDistribution	<input checked="" type="radio"/> 44.88%	Weibull ($\lambda: 56.010013, \eta: 1.673703$)					
	<input type="radio"/> 23.7%	Erlang ($\lambda: 26.379431, \kappa: 1.914421$)					
	<input type="radio"/> 2.47%	LogN ($\mu: 3.638642, \sigma: 0.918553$)					
	<input type="radio"/> 1.06%	Exp ($\lambda: 50.50133$)					

[Manage distributions](#) [Mix distributions](#) [Export](#) [Import](#) [Import via API](#)

[Save](#)

Each distribution in the set has a goodness-to-fit value (GTF) which 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 action column.

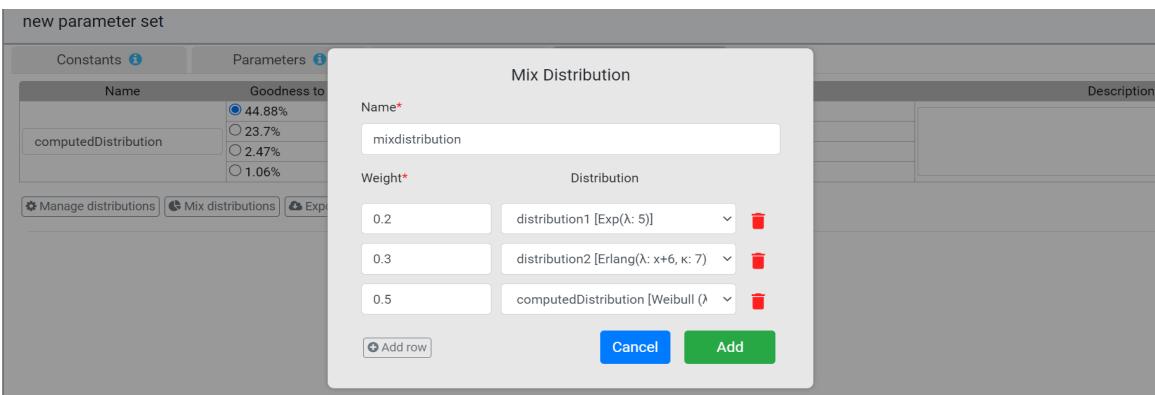


Constants ⓘ	Parameters ⓘ	Failure distributions ⓘ	Empirical failure dist. ⓘ	Description	Action
Name	Goodness to fit	Distribution			
computedDistribution	<input checked="" type="radio"/> 44.88%	Weibull ($\lambda: 56.010013, \eta: 1.673703$)			 
	<input type="radio"/> 23.7%	Erlang ($\lambda: 26.379431, \kappa: 1.914421$)			 
	<input type="radio"/> 2.47%	LogN ($\mu: 3.638642, \sigma: 0.918553$)			 
	<input type="radio"/> 1.06%	Exp ($\lambda: 50.50133$)			 
computedDistribution1	<input checked="" type="radio"/> 44.88%	Weibull ($\lambda: 56.010013, \eta: 1.673703$)			 
	<input checked="" type="radio"/> 23.7%	Erlang ($\lambda: 26.379431, \kappa: 1.914421$)			 
	<input type="radio"/> 2.47%	LogN ($\mu: 3.638642, \sigma: 0.918553$)			 
	<input type="radio"/> 1.06%	Exp ($\lambda: 50.50133$)			 
mixDistribution	<input checked="" type="radio"/> 99.92%	Weibull ($\lambda: 55.272241, \eta: 1.619002$)			 
	<input type="radio"/> 18.04%	Erlang ($\lambda: 22.67194, \kappa: 2.183942$)			 
	<input type="radio"/> 0%	Exp ($\lambda: 49.514204$)			 
	<input type="radio"/> 0%	LogN ($\mu: 3.65618, \sigma: 0.795911$)			 

[Manage distributions](#) [Mix distributions](#) [Export](#) [Import](#) [Import via API](#)

[Save](#)

- 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, $d3 = 0.3*d1 + 0.7*d2$, where d1 and d2 are existing empirical failure distributions.



- 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 and use them in other projects. You can also import and export the whole parameter set.

Metrics

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

Metrics			
Basic	Complex	Criticality	Custom
Name	Formula	Parameters	Labels
Mean time to failure (MTTF)	$T=?[F \text{ topLevelEvent}]$	No parameters	topLevelEvent
Reliability	$1 - P=?[F \leq \text{time_bound} \text{ topLevelEvent}]$	time_bound	topLevelEvent
Average failure probability per unit time (AFH)	$1/\text{time_bound} * P=?[F \leq \text{time_bound} \text{ topLevelEvent}]$	time_bound	topLevelEvent

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

- i. Mean-time-to-failure. Expected time to system failure or scenario occurrence.
- ii. Reliability: Probability of failure in a given time bound.
- iii. Unreliability: The complement of reliability (1- Reliability).
- iv. Average failure probability per hour.

Complex. It contains six metrics. These metrics cannot be verified directly by the Storm model-checker. They need some additional computations at the back end for their verification.

- v. 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.
- vi. 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.
- vii. 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 to reach this state while not being degraded before.
- viii. 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.
- ix. Failure under Limited Operation in Degradation (FLOD) 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.
- x. 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).

Metrics			
Basic	Complex	Criticality	Custom
Name	Formula	Parameters	Labels
Full Function Availability (FFA)	$1 - P=? [F \leq time_bound \text{ topLevelEvent} \mid degraded]$	time_bound	topLevelEvent, degraded
Failure without degradation (FWD)	$P=? [(degraded) U \leq time_bound (\text{topLevelEvent} \& !degraded)]$	time_bound	topLevelEvent, degraded
Mean time from degradation to failure (MTDF)	$\sum_{s \in \text{degraded}} (P=? [(degraded \ U s)] * T=? [F \text{ topLevelEvent}])$	No parameters	topLevelEvent, degraded
Minimal degraded reliability (MDR)	$\text{argmin}_{s \in \text{degraded}} (1 - P=? [F \leq time_bound \text{ topLevelEvent}])$	time_bound	topLevelEvent, degraded
Failure under limited operation in degradation (FLOD)	$\sum_{s \in \text{degraded}} (P=? [(degraded) U \leq time_bound s] * P=? [F \leq drive_cycle \text{ topLevelEvent}])$	time_bound, drive_cycle	topLevelEvent, degraded
System integrity under limited fail-operation (SILFO)	$1 - (FWD + FLOD)$	time_bound, drive_cycle	topLevelEvent, degraded

Criticality. It contains only the Birnbaum Index (BI) at the moment – a common way to measure the sensitivity of the system to an element is the Birnbaum importance index.

Metrics			
Basic	Complex	Criticality	Custom
Name	Formula	Parameters	Labels
Birnbaum Index (BI)	$(P=? [F \leq time_bound component_failed] / P=? [F \leq time_bound component_failed]) * (P=? [F \leq time_bound topLevelEvent \& component_failed] / P=? [F \leq time_bound component_failed] - P=? [F \leq time_bound topLevelEvent \& !component_failed] / P=? [F \leq time_bound !component_failed])$	time_bound	topLevelEvent, component_failed

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

Custom Metric i

Name* i

Enter parameter and label names (comma separated) to be used in the below formula.

Parameters i

Labels i

Formula* i

Description

Cancel
Add

- xi. 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.
- xii. The formula can be defined using probabilistic computation tree logic (PCTL)/continuous stochastic logic (CSL). For example, $P = ? [\text{true} \cup \text{failed} \leq 10000 \wedge ! \text{mode1}]$, where failed and mode1 represent quantifiable states. The grammar of expressions is given [here](#).
- xiii. Note. The parameters given on the above screen are exclusively dedicated to metrics. Their values cannot be taken from the parameter set that is attached with a failure model at the time of analysis. However, their values can be changed at the time of analysis, and plots can be drawn for metric values.

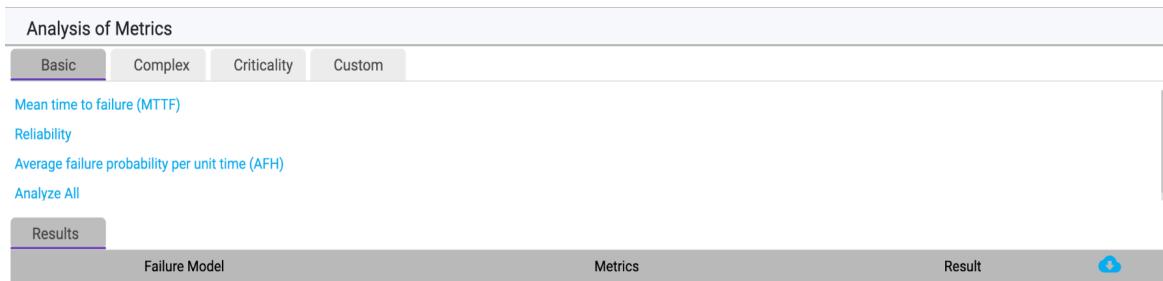
System analysis with SAFEST

In the SAFEST tool, different types of analysis – from basic to complex – can be performed for each failure model in the project.

1. (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 usually 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

Analysis

Metrics *

Minimal degraded reliability (MDR): $\operatorname{argmin}_s \in \text{degraded} (1 - P = ? [F \leq \text{time_bound}] \text{topLevelEvent})$

Failure model*

OWT

Metric parameters

Name	Value <small>i</small>
time_bound	100

Assign quantifiable-state labels (of the model) to metric labels

Metric label	Quantifiable-state label
topLevelEvent	failed (Root Element)
degraded	failed (Root Element)

Model parameter set i

parameter_set1

Constants

Name	Value <small>i</small>
W1	0.000039
W2	0.000008

Output tab: Existing New

Results

Cancel Run

- In the “Failure model” dropdown, a model that is selected as a default model in the “Failure Models” window is automatically selected.
- The time-bound (life cycle) parameter of Metric is assigned a value that is entered at the time of failure model creation.

Model Information

Name*	Version	Author
Test_Project_model	1.0	John
Fault tree	<input type="radio"/> Static <input checked="" type="radio"/> Dynamic	
Time bound (Life cycle)* <small>i</small>	100	Parameter set <small>i</small>
Description		
Save		

- Give values to metric parameters. Note that metric parameters cannot take values defined in the parameter set attached with the model.
- Assign each label in the metric (which represents a class of states) a quantifiable-state (which indicates a class of model sub-scenarios) of the model.
- A parameter set which is attached with the selected failure model (above) is automatically selected. It can be changed at this point.
- Optionally, change the values of constants defined in the parameter set which is selected. Note that values of other elements in the parameter set (real-value expressions, (empirical) failure distributions) cannot be changed at the time of analysis.
- Select analysis method either Markov or BDD. Note that BDD analysis does not work for all of the metrics.
- Finally, select a tab on which result of the analysis has to be displayed.

Analysis of Metrics

Basic	Complex	Criticality	Custom																					
Mean time to failure (MTTF)	Reliability	Unreliability																						
Average failure probability per unit time (AFH)	Analyze All																							
<table border="1"> <thead> <tr> <th colspan="2">Results</th> <th>Failure Model</th> <th>Metrics</th> <th>Information</th> <th>Result</th> <th><small>Cloud</small></th> </tr> </thead> <tbody> <tr> <td>OWT</td> <td>Reliability</td> <td colspan="3">Failure model: OWT Type: Markov Analysis Parameter set: parameter_set1 time_bound: 100 Quantifiable States: topLevelEvent: failed (Root Element)</td> <td>0.845321</td> <td><small>Download</small> <small>Open</small> <small>Red</small></td> </tr> <tr> <td>OWT</td> <td>Mean time to failure (MTTF)</td> <td colspan="3">Failure model: OWT Type: Markov Analysis Parameter set: parameter_set1 Quantifiable States: topLevelEvent: failed (Root Element)</td> <td>593.60803</td> <td><small>Download</small> <small>Open</small> <small>Red</small></td> </tr> </tbody> </table>				Results		Failure Model	Metrics	Information	Result	<small>Cloud</small>	OWT	Reliability	Failure model: OWT Type: Markov Analysis Parameter set: parameter_set1 time_bound: 100 Quantifiable States: topLevelEvent: failed (Root Element)			0.845321	<small>Download</small> <small>Open</small> <small>Red</small>	OWT	Mean time to failure (MTTF)	Failure model: OWT Type: Markov Analysis Parameter set: parameter_set1 Quantifiable States: topLevelEvent: failed (Root Element)			593.60803	<small>Download</small> <small>Open</small> <small>Red</small>
Results		Failure Model	Metrics	Information	Result	<small>Cloud</small>																		
OWT	Reliability	Failure model: OWT Type: Markov Analysis Parameter set: parameter_set1 time_bound: 100 Quantifiable States: topLevelEvent: failed (Root Element)			0.845321	<small>Download</small> <small>Open</small> <small>Red</small>																		
OWT	Mean time to failure (MTTF)	Failure model: OWT Type: Markov Analysis Parameter set: parameter_set1 Quantifiable States: topLevelEvent: failed (Root Element)			593.60803	<small>Download</small> <small>Open</small> <small>Red</small>																		

- The results can be downloaded as a csv file for each tab separately.

2. Bounded analysis

In order to compute exact results for measures, first 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:

Bounded Analysis

Metric

Mean time to failure (MTTF) : T=? [F topLevelEvent]

Failure Model

OWT

Assign quantifiable-state labels (of the model) to metric labels

Metric label	Quantifiable-state label
topLevelEvent	failed (Root Element)

Parameter set* i

parameter_set1

Constants

Name	Value
W1	0.000039
W2	0.000008

Error margin between upper and lower bound of the actual value

0 %

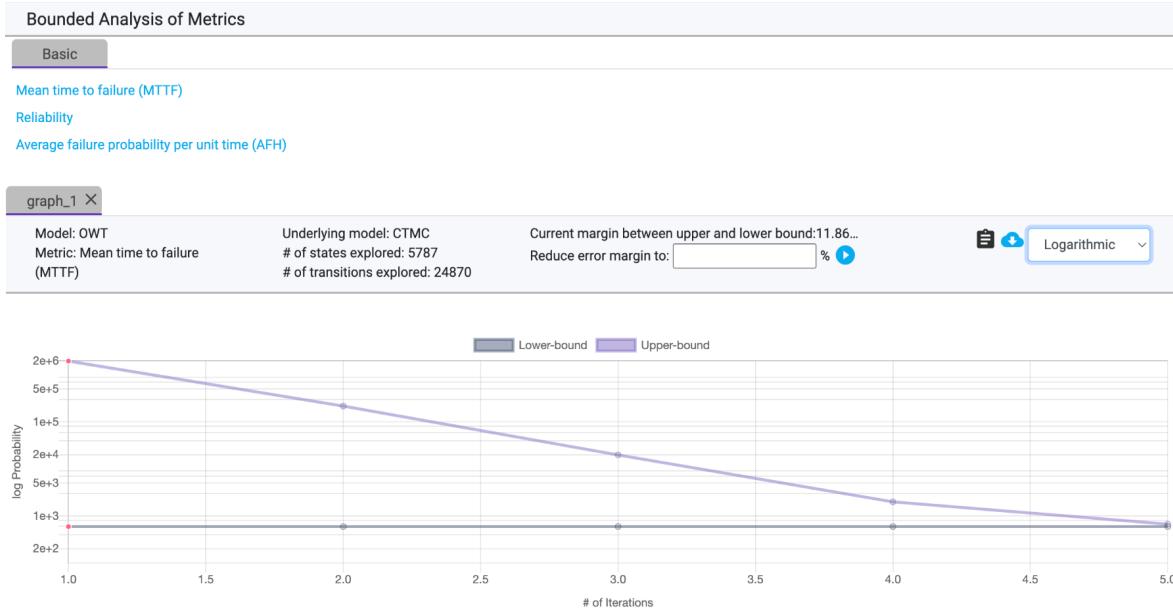
Graph name* i **Y-axis label***

graph_1

Probability

Cancel
Run

- All fields are filled up as described in the “Analysis” case with few additions:
- Enter acceptable error margin between upper and lower of the actual value.
- Optionally enter graph name and the label of its Y-axis. Note X-axis will always represent the number of iterations in this case.



- 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 one is interested to further reduce the error margin, it will be insert a new value in the text field and click the play button .
- One can apply a log function on the values of Y-axis by selecting it on the right side of the graph.
- The graph values can be downloaded by clicking on the download icon.

3. Graphs

Reliability measures help figuring out optimal maintenance schedules of systems thus reducing their downtimes and saving cost at the same time. Fault trees that model sub-systems of systems can even predict their health individually thus helping make even more detailed maintenance schedules. We provide a graphical interface to plot and compare measures of interest e.g. reliability of different sub-systems, which is helpful in deciding maintenance schedules.

Click on the “Graphs” link under “Computing” in the left panel and then click “Reliability”. The following window will appear:

Graph

Metric

Reliability : $1 - P=? [F \leq time_bound \text{ topLevelEvent}]$

Failure model

OWT

Metric Parameters

Name	Single point		Range			
	Value	Start	End	Step		
time_bound	<input checked="" type="radio"/>	100	<input type="radio"/>	1	100	1

Assign quantifiable-state labels (of the model) to metric labels

Metric label	Quantifiable-state label
topLevelEvent	failed (Root Element)

Model parameter set* i

parameter_set1

Constants

Name	Single point		Range			
	Value	Start	End	Step		
W1	<input checked="" type="radio"/>	0.000039	<input type="radio"/>	1	0.000039	1
W2	<input checked="" type="radio"/>	0.000008	<input type="radio"/>	1	0.000008	1

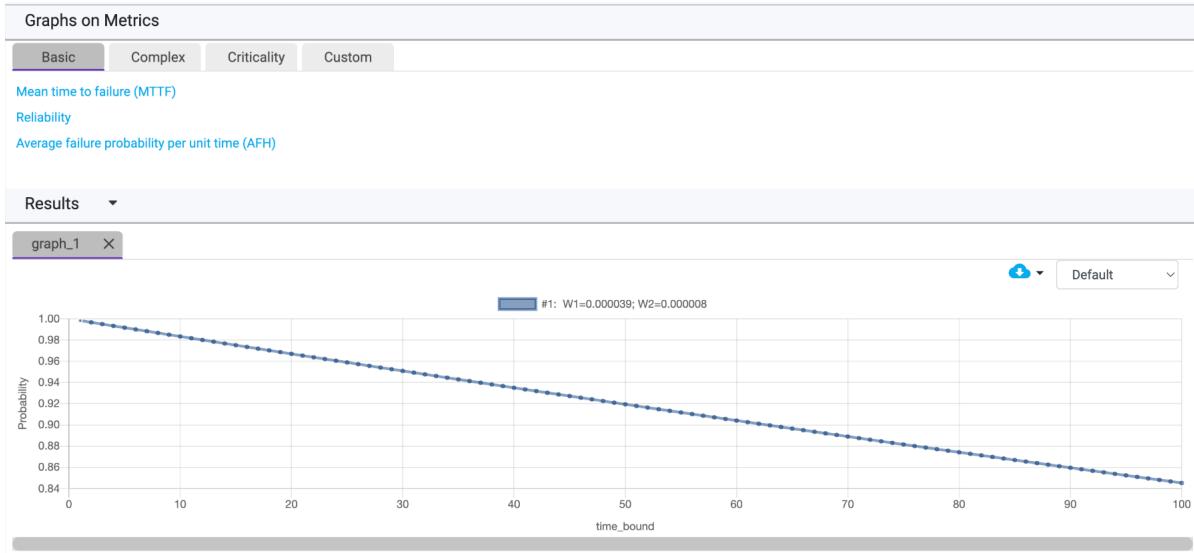
Analysis type: Markov BDD

Graph New Existing

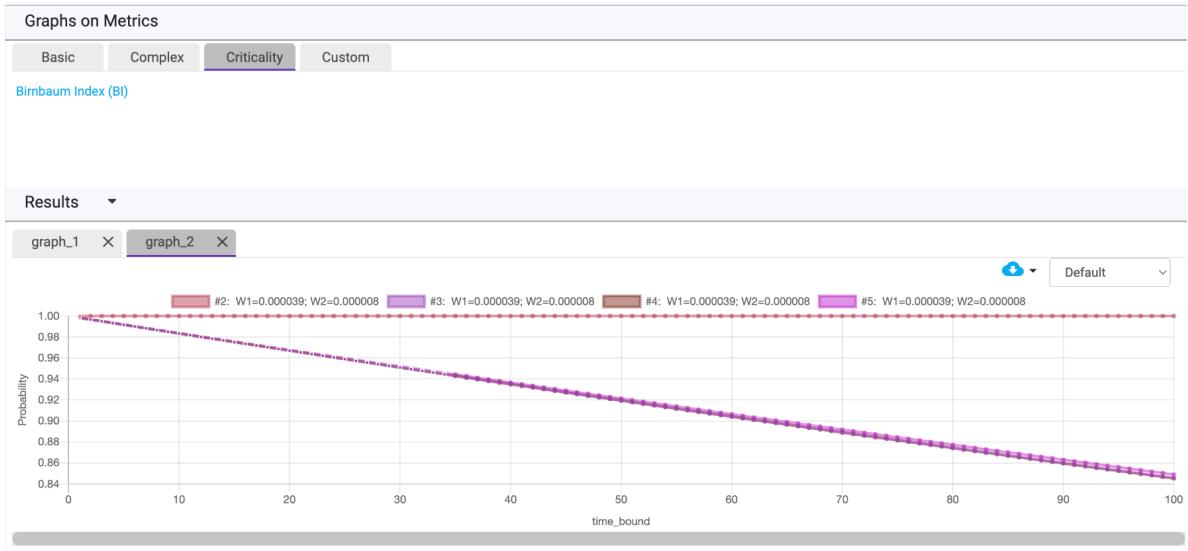
Name* i Variable on X-axis Y-axis label*

Cancel Run

- One can specify a range of values of Metric parameters and Constants defined in the parameter set which is selected above.
- A graph can be plotted on an existing graph as well that has the same variable on X-axis.
- The variable on the X-axis of the graph can be specified either from the Metric parameters or Constans in the parameter set which is selected above.



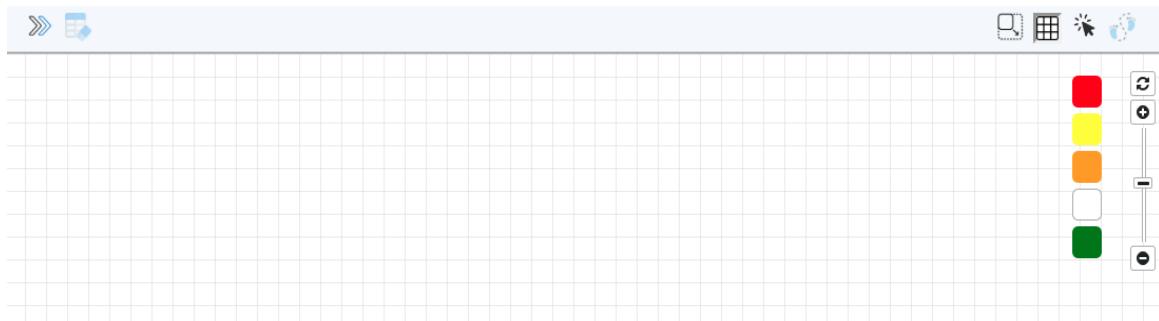
- In case of Birnbaum Index (on Criticality tab), one can draw plot for multiple components at the same time as:



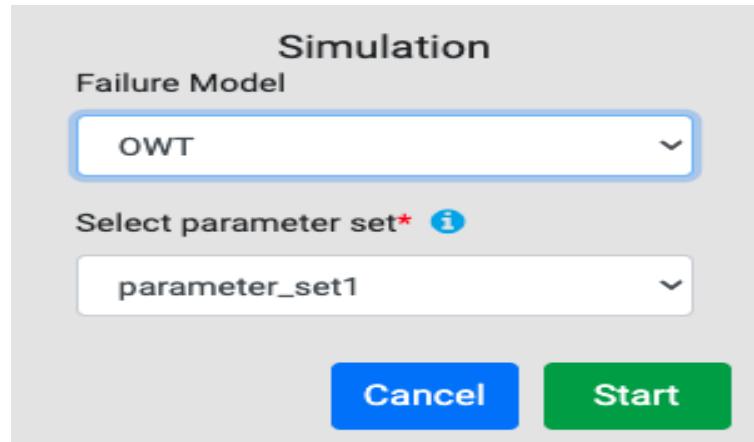
4. Interactive simulation

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

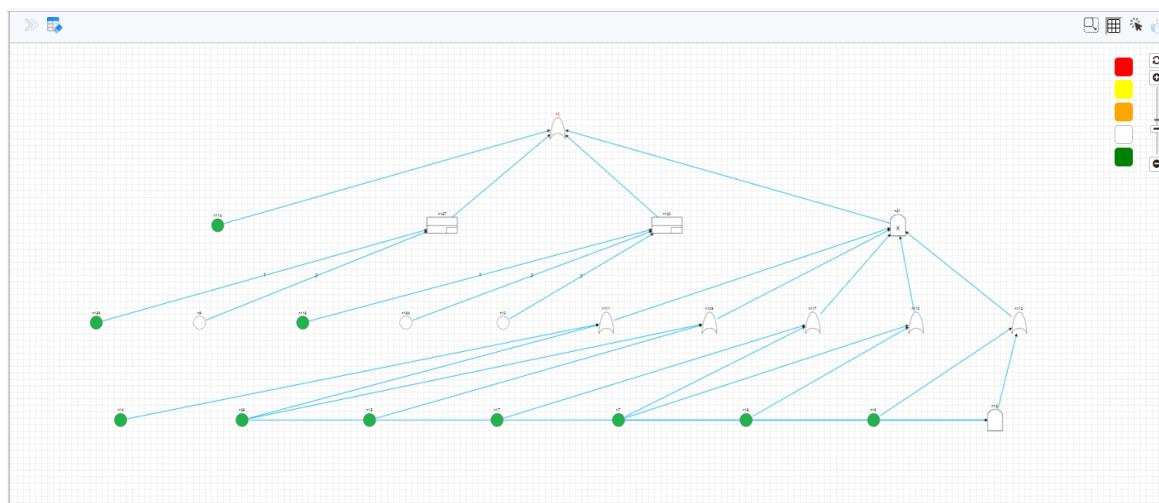
Click on the “Interactive Simulation” link under “Computing” in the left tab. The following screen will appear.



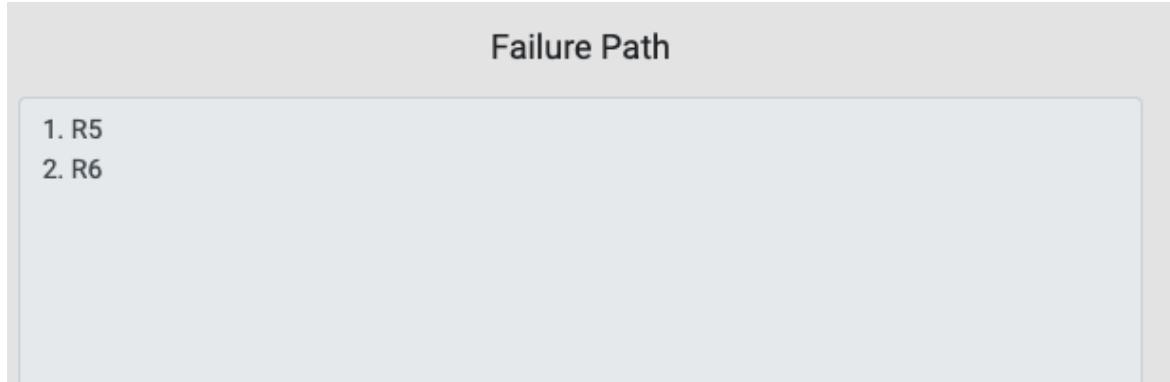
Click on the icon ➤ to start the simulation. User will be prompted to select a failure model and a parameter set as:



On clicking “Start” the following screen appears:



- All basic events (BEs) that can fail are shown as green.
- User clicks any green BE to fail it. Its color will be turned into Red. After this, BEs which are operational and cannot fail remain White, those which are in fail-safe state are Orange, those which are in dont-care state are Yellow.
- 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 click on the icon  as:

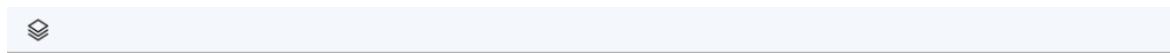


- User can restart simulation by clicking on the icon .

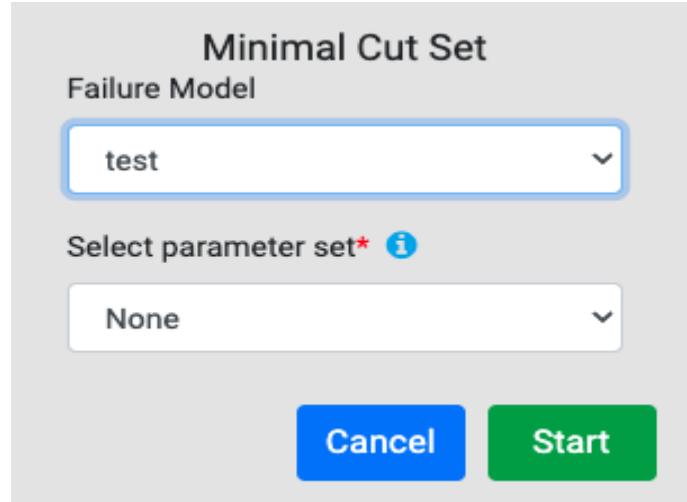
5. Minimal cut set (for static fault trees)

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.

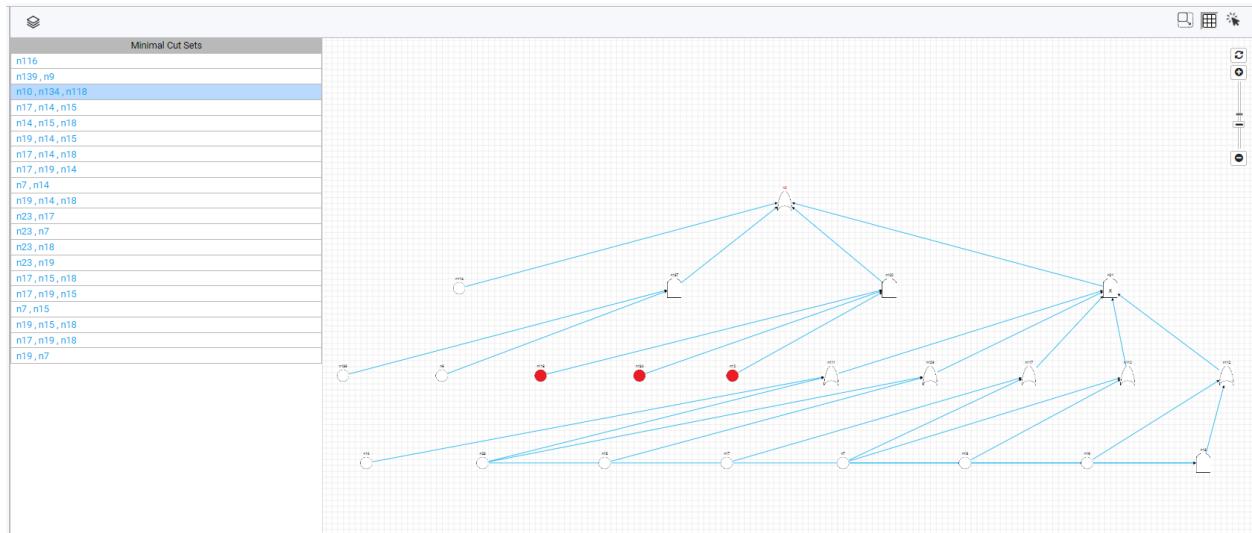
Click on the “Minimal cut set” link under “Computing” in the left tab. The following screen will appear.



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



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



- All minimal cut sets will be shown on the left of the screen.
- On clicking a cut set, the corresponding BEs will be highlighted (in Red) in the tree.

Main Toolbar

1) File Menu:

- New Project: it starts a process to generate a system failure model from scratch.
- Open Project. It opens an already existing system failure model.
- Open Recent Project: It'll restart a recently finished project. Even if the SAFEST crashes for any reason, a project is still in the working folder and can be accessed again.

- d) Export Project. It exports the current project in a file with .safest extension.
- 2) View:
- a) Simple View. It is for simple users. Under this view, a user cannot create quantifiable states as well as define custom metrics.
 - b) Advance View. It is for advanced users or researchers. This view gives the full functionality of the SAFEST tool.
- 3) Help:
- a) Documentation: It contains the grammar for expressions.
 - b) Activation key: Here you can add a license key and activate SAFEST tool functionality.

Grammar:

Expressions detail

- Identifier (id):
 - started with a capital and small letters(a-z A-Z) followed by underscores, letters, or numbers (a-z A-Z 0-9)
- Mode Name (mode):
 - started with a capital, and small letters(a-z A-Z) followed by underscores, letters, dot, or numbers (a-z A-Z . 0-9)
- 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

Real expressions

- Keywords:
 - [pow, log]
- Context Free Grammar:
 - $E \rightarrow E \text{ OP } E \mid nc \mid id \mid (E) \mid \text{pow}(E,E) \mid \text{log}(E,E)$
 - $\text{OP} \rightarrow + \mid - \mid * \mid /$

Boolean Logic

- Keywords:
 - No keywords
- Context Free Grammar:
 - $E \rightarrow E \text{ OP } E \mid mode \mid (E) \mid !E$
 - $\text{OP} \rightarrow \mid \mid \&$

Continuous Stochastic Logic

- Keywords:

- [true,false,Pmin,Pmax,Smin,Smax,Tmin,Tmax,LRAmin,LRAmax,P,R,T,S,LRA,min,max,G,U,F,W,C,I,failed]
- Context Free Grammar:
 - $\text{PROP} \rightarrow P \text{ OP2 TYPE} [\text{PathFormula}] | T \text{ OP2 TYPE} [\text{RewardFormula}] | \text{LongRun OP2 TYPE} [\text{StateFormula}]$
 - $\text{TYPE} \rightarrow =? | \text{OP3 E}$
 - $\text{LongRun} \rightarrow \text{LRA} | S$
 - $\text{PathFormula} \rightarrow \text{OP4 BoundedExpression StateFormula} | \text{StateFormula OP5 BoundedExpression StateFormula}$
 - $\text{BoundedExpression} \rightarrow ^\wedge \{ \text{Bound} \} | \{ \text{Bound} \} | \text{Bound} | \text{null}$
 - $\text{Bound} \rightarrow [E,E] | \text{OP3 TIME}$
 - $\text{TIME} \rightarrow (E) | \text{nc}$
 - $\text{RewardFormula} \rightarrow I = E | C \leq E | F \text{ StateFormula} | \text{LongRun}$
 - $\text{StateFormula} \rightarrow \text{StateFormula OP6 StateFormula} | P \text{ OP2 OP3 E} [\text{PathFormula}] | \text{LongRun OP2 OP3 E} [\text{StateFormula}] | \text{mode} | \text{failed} | (\text{StateFormula}) | \text{true} | !\text{StateFormula}$
 - $\text{OP} \rightarrow =? | & | || = | != | \leq | \geq | > | < | + | - | * | / | %$
 - $\text{OP1} \rightarrow + | -$
 - $\text{OP2} \rightarrow \text{min} | \text{max} | \text{null}$
 - $\text{OP3} \rightarrow \leq | \geq | > | <$
 - $\text{OP4} \rightarrow G | F$
 - $\text{OP5} \rightarrow U | W | R$
 - $\text{OP6} \rightarrow || | \&$
 - $E \rightarrow E \text{ OP } E | \text{id} | \text{nc} | (E) | !(E) | (\text{OP1 nc})$