Mercury Tool Manual
v4.8

MoDCS Research Group

[http://www.modcs.org]

CIn - Centro de Informatica UFPE

Cidade Universitaria - 50740-540 - Recife - Brazil

- Tel +55 81 2126.8430 -

March 13, 2020

Contents

1 Overview

This manual describes the **Mercury** tool: a software for supporting performance, dependability, and energy flow modeling in an easy and powerful way. The tool provides graphical interfaces for creating and evaluating Stochastic Petri Nets (SPNs), Continuous-Time Markov Chains (CTMCs), Discrete-Time Markov Chains (DTMCs), Reliability Block Diagrams (RBDs), Fault Trees (FTs), and Energy Flow Models (EFMs).

Mercury has been developed by **MoDCS** (Modeling of Distributed and Concurrent Systems) Research Group at Informatics Center (CIn) of the Federal University of Pernambuco (UFPE) in Brazil since 2009. A comprehensive view of features is described here as well as steps for creating, editing, and evaluating the models supported by the tool. An overview of the Mercury features is depicted below:

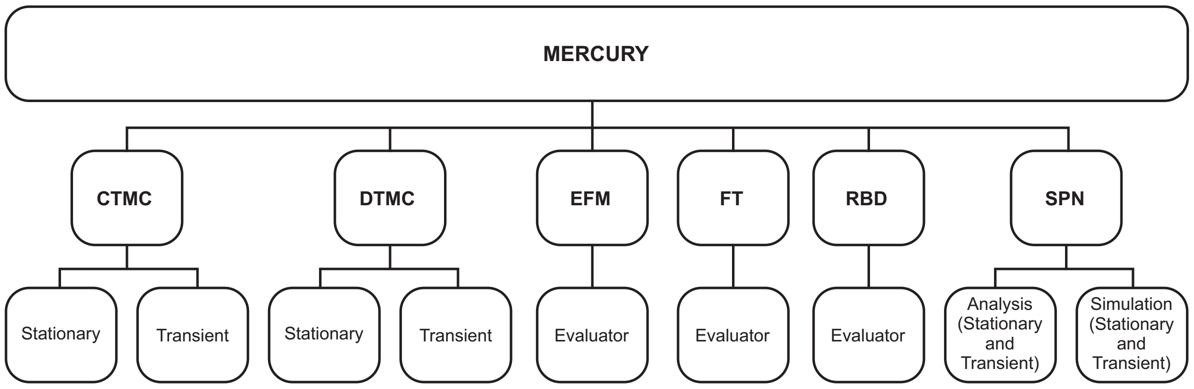


Figure 1: Mercury Tool - Features

Mercury has been developed in Java language, which provides platform independence. The graphical interface allows one to model systems by using one or more views — RBD, FT, EFM, SPN, CTMC, or DTMC — whereas auxiliary tools (e.g., random variate generation and moment matching) are also available. Thus, users can adopt the most appropriate view according to their needs.

In addition, Mercury tool also provides a function that allows importing models built in other engines that adopt the ".TN" standard format (used in tools such as TimeNet [1]). Moreover, there is also an option to export the models created on Mercury to a ".TN" file that is compliant to that standard. All projects developed in Mercury are saved in a ".xml" file that stores all information related to the created models.

1.1 How to Install the Tool

In order to install the newest version of the Mercury tool, the user needs to access the URL `http://www.modcs.org/mercury`. There is a license agreement document that must be signed and sent to the Mercury creators before the access to the download page is granted to the user.

Mercury is made available in a compressed archive, containing the executable files (.jar and .bat), a folder containing the third-party libraries required for Mercury execution, and a folder with example models. All files require approximately 80 MB of disk space. The memory footprint of Mercury in runtime is about 60 MB, but it might increase depending on the size of models and type of analysis executed with the tool. Extract the archive, and double-click the executable file (**Mercury.jar** or **Mercury.bat**) file to open the Mercury tool. The **lib** folder must be kept in the same path of the jar file.

1.1.1 Linux System Requirements

This section specifies the minimum Linux system configuration required by the Fault Tree module. Ensure that the system meets these minimum requirements: 1) Java Runtime Environment (JRE) or Java Development Kit (JDK) in version 1.8+; and 2) OpenJFX package. On Ubuntu, the admin may use the following command to have OpenJFX installed: *sudo apt-get install openjfx*. Fault Tree module will not be available if both requirements are not met.

1.2 Graphical User Interface (GUI)

Mercury provides six different views: (i) SPN, (ii) CTMC, (iii) DTMC, (iv) RBD, (v) EFM, and (vi) FT. This section shows each one of these views, briefly describing them. Each formalism has its own section and more details regarding each view are available in its respective section.

1.2.1 SPN View

Regarding SPNs, the Mercury tool makes it possible to carry out evaluations through simulation or analysis (i.e., numerical solution of the underlying Markov chain). Both kinds of evaluation allow computing **transient** and **stationary** metrics. Time-dependent metrics are obtained through transient evaluations while steady-state metrics are obtained through stationary evaluations. Figure 2 depicts the SPN View with an SPN as an example. See Section 2 for further information about the support provided by Mercury for the SPN formalism.

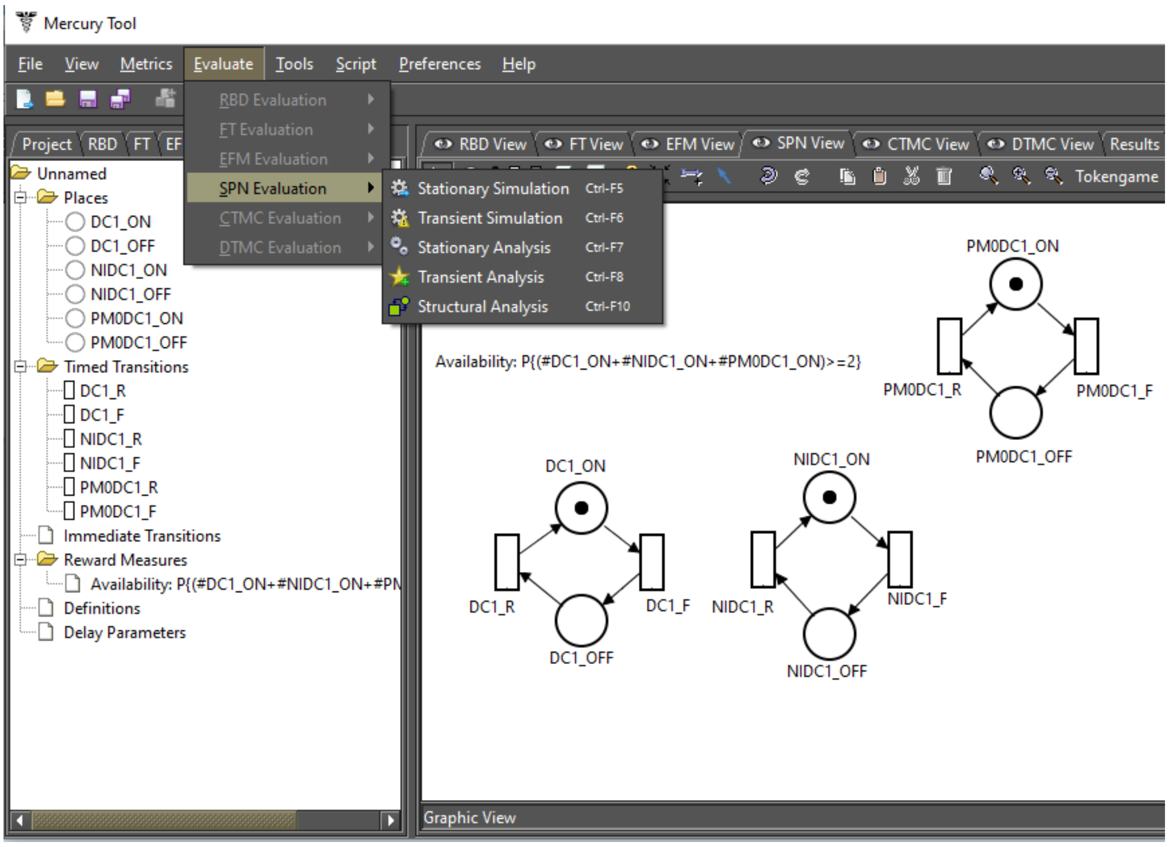


Figure 2: SPN View

1.2.2 CTMC View

The CTMC view (see Figure 3) provides features for drawing and evaluating continuous-time Markov chains. The numerical solution of CTMCs may be carried out through stationary or transient analysis. For computing stationary metrics, two methods are available: GTH (Grassmann-Taksar-Heyman) and Gauss-Seidel. Transient metrics are obtained through the "Uniformization" method (also known as Jensen's method) as default, but the user might also try the "4th-order Runge Kutta" method. Sensitivity analysis is also available at the CTMC view. The rate of each transition might be defined by means of polynomial expressions with user-defined variables (named as parameters on Mercury). Parameter names can include Greek letters. Besides states and transitions, users can define reward rates assigned to the states. In such a case, CTMCs become Markov reward models. For models having absorbing states, Mercury also enables computing probability of absorption and mean time to absorption. The user may create custom metrics by formulating expressions that may include state probabilities. Parameters and metrics are easily viewed and modified in the CTMC editor. See Section 4 for further information about the support provided by Mercury for the CTMC formalism.

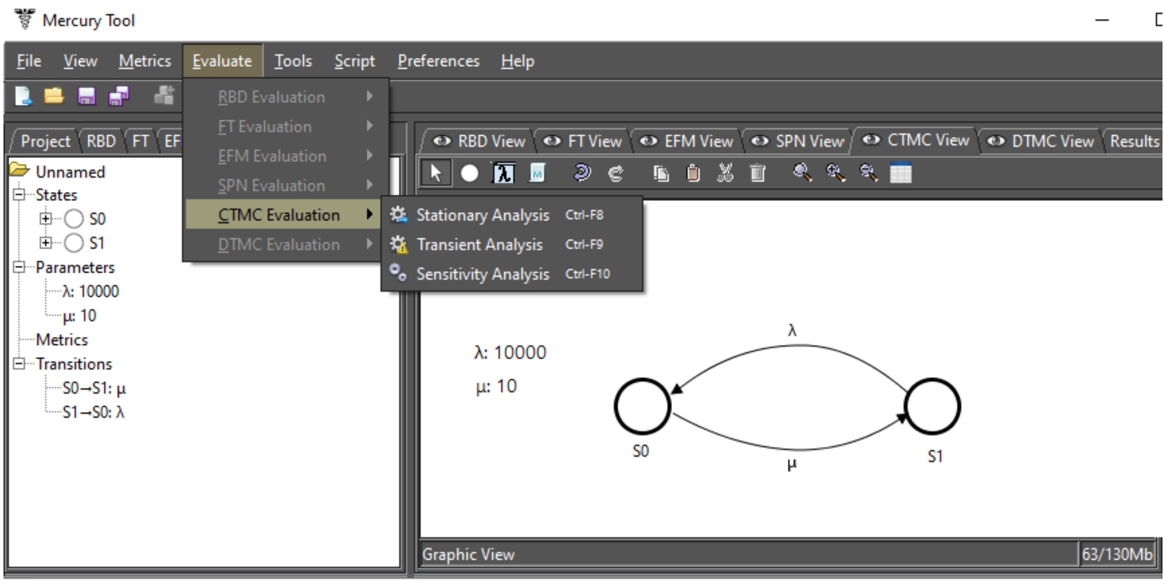


Figure 3: CTMC View

1.2.3 DTMC View

The DTMC view (see Figure 4) provides features for drawing and evaluating discrete-time Markov chains. The numerical solution of DTMCs may be carried out through stationary or transient analysis. For computing stationary metrics, two methods are available: GTH (Grassmann-Taksar-Heyman) and Gauss-Seidel. Parameter names can include Greek letters. For models that have absorbing states, Mercury also enables computing probability of absorption and mean time to absorption. The user may create custom metrics by formulating expressions that may include state probabilities. Parameters and metrics are easily viewed and modified in the DTMC editor. See Section 5 for further information about the support provided by Mercury for the DTMC formalism.

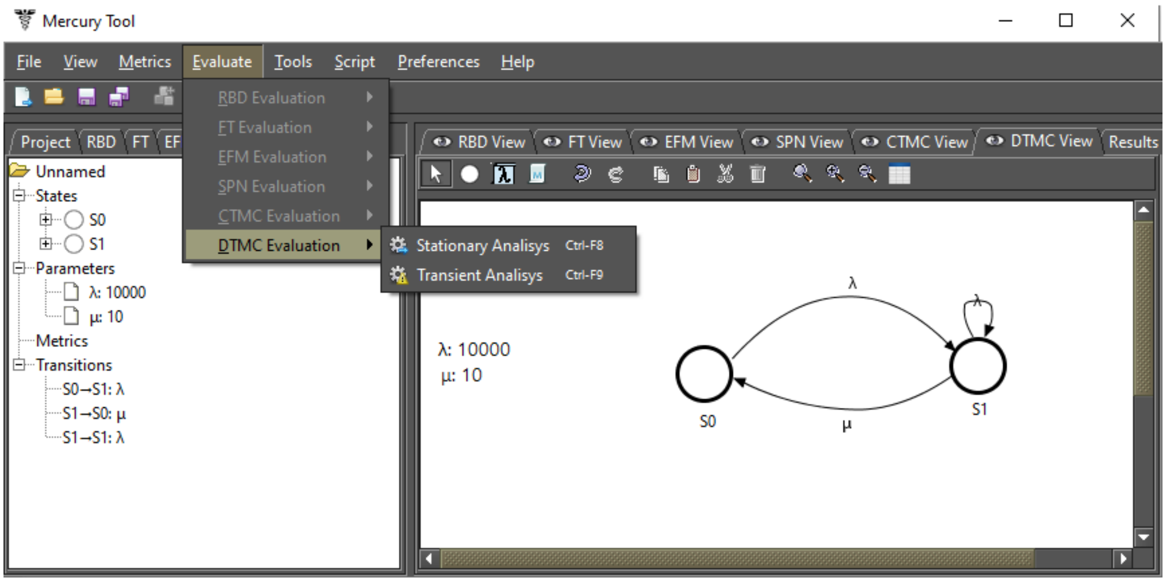


Figure 4: DTMC View

1.2.4 RBD View

Reliability Block Diagram (RBD) is a success-oriented modeling approach and makes it possible to create a visual representation of a system that shows how the components contribute to the failure or success of the system. The RBD view (see Figure 5) provides features for performing reliability and availability analysis on large and complex systems using block diagrams. The types of configurations supported by the tool are series, parallel, and k-out-of-n. It also provides the solution by closed-form equations so that the results are usually obtained faster than simulation or numerical solutions of other models. In addition, users can add labels and carry out experiments for a given component individually. When creating a new project, the default RBD model is created with an empty block. In the RBD view, the default model presents the empty block named b1. This block is gray and represents that its properties have not yet been defined. See Section 3 for further information about the support provided by Mercury for the RBD formalism.

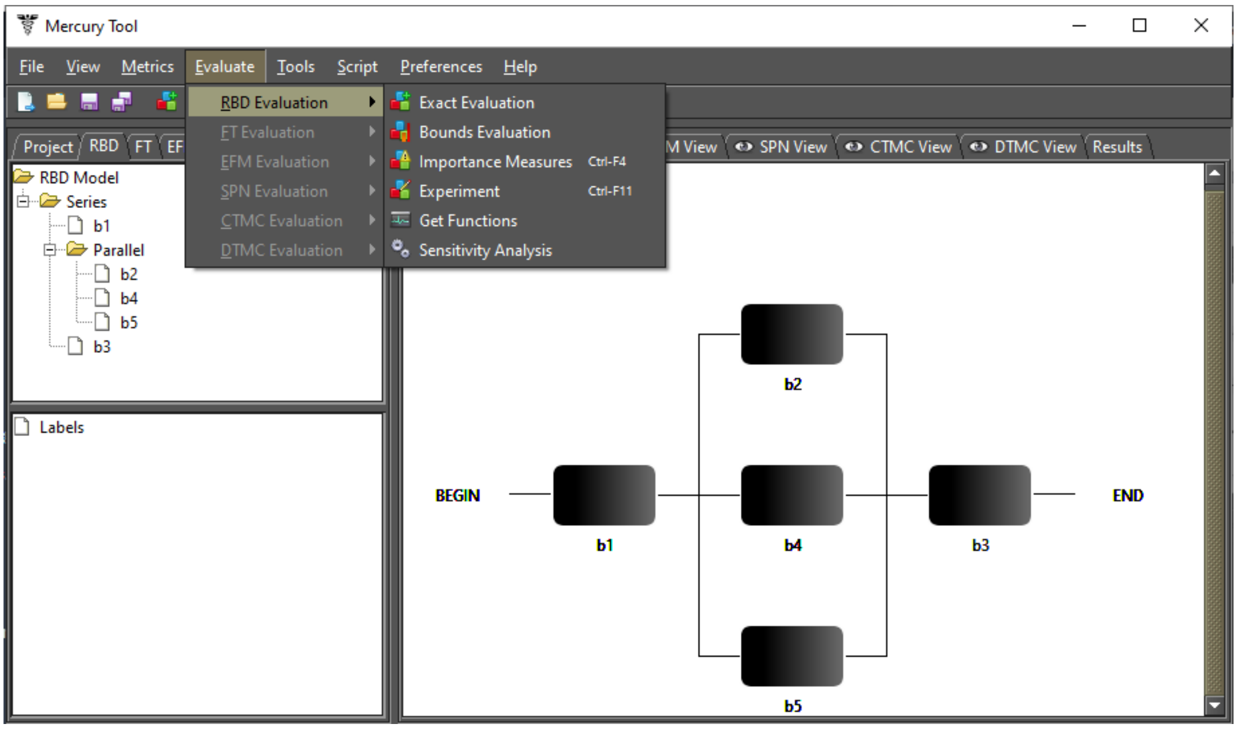


Figure 5: RBD View

1.2.5 EFM View

The EFM view provides features for computing the sustainability and cost estimates of data center power and cooling infrastructures while conforming to the energy constraints of each device. The EFM models represent the energy flow between system components in terms of the respective efficiency and the maximum energy that each component can provide (for electrical devices) or the maximum cooling capacity (for cooling devices). Figure 6 depicts an example of an EFM model. See Section 6 for further information about the support provided by Mercury for the EFM formalism.

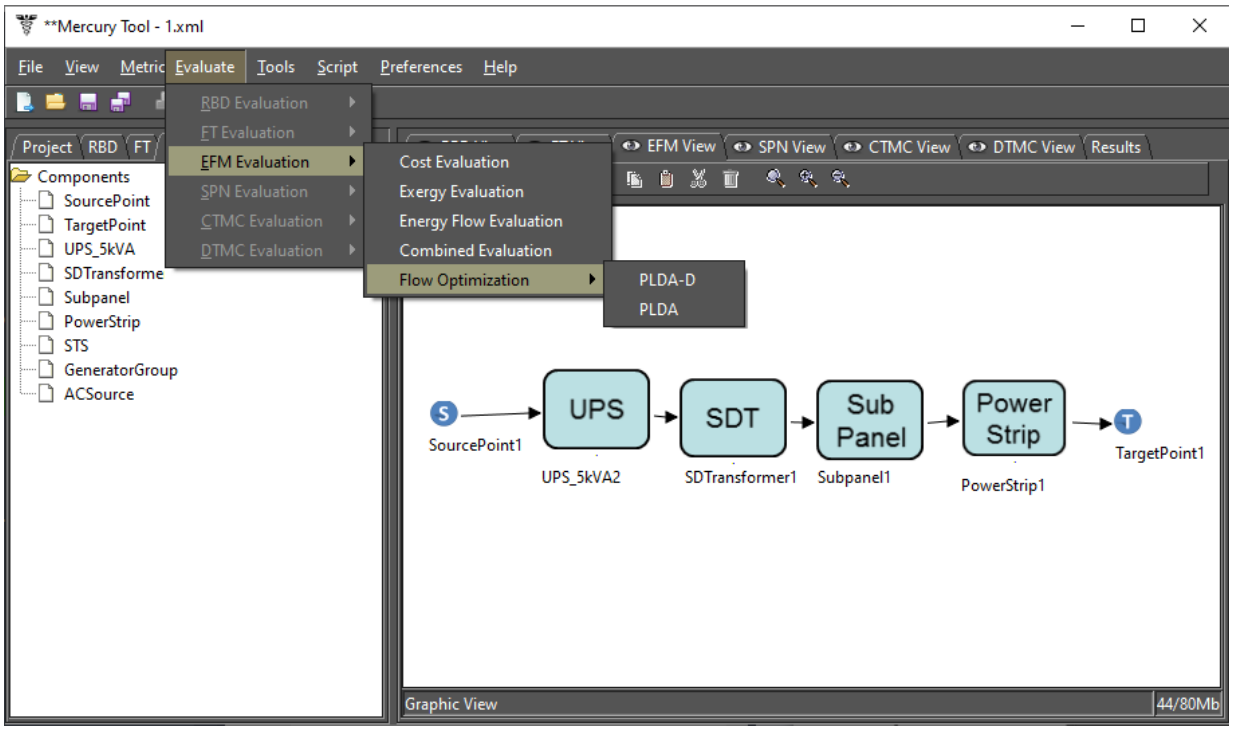


Figure 6: EFM View

1.2.6 FT View

Fault Trees (FTs) and RBDs differ from each other in their final purpose. FT is a top-down logical diagram and it makes possible to create a visual representation of a system that shows the logical relationships between the associated events and causes lead that lead the system to fail. When creating a new project, the default FT model is created with an empty top event (see Figure 7). In the FT view, the default model presents a top event named **FAILURE**. This top event has the word undefined as a description since no event leads to it. See Section 7 for further information about the support provided by Mercury for the FT formalism.

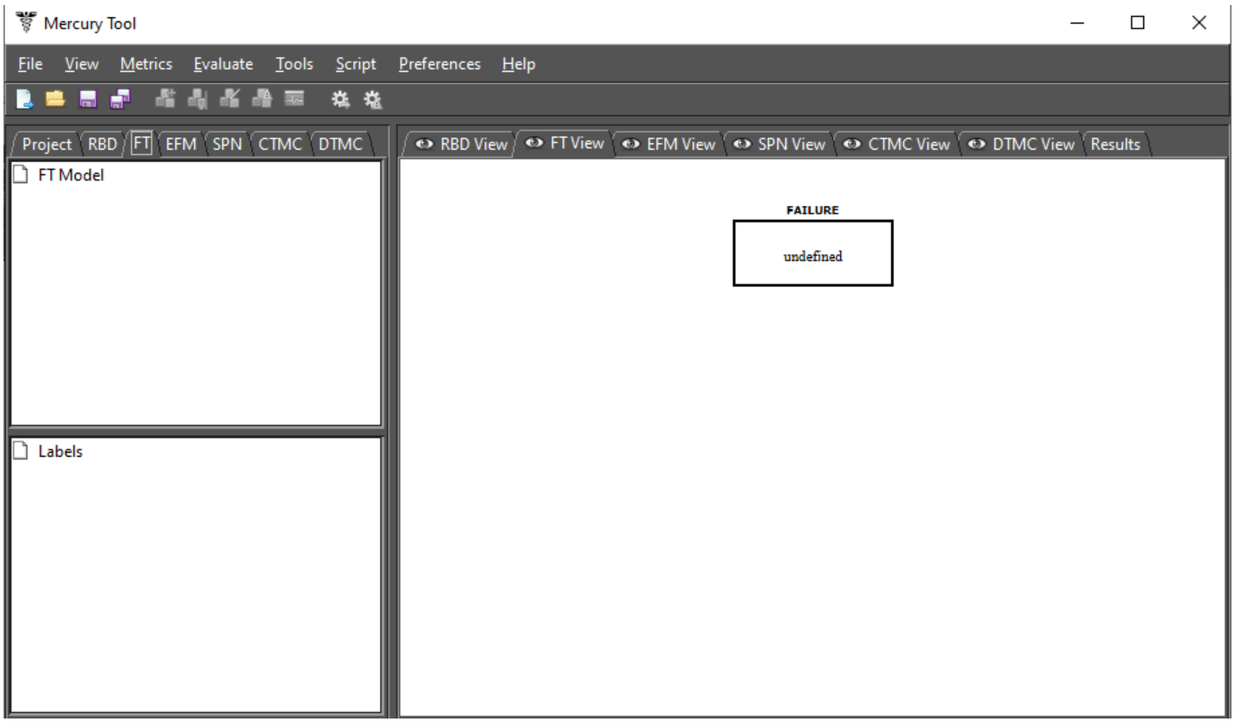


Figure 7: FT View

To make the Fault Tree module available on Linux-based distributions, it is necessary to install JavaFX. It can be downloaded by accessing this URL `https://www.oracle.com/technetwork/pt/java/javafx/downloads/index.html` or installed through a Linux terminal. For Microsoft Windows systems, there is no need to install any additional packages.

1.3 Main Menu

In this section, we will present the main menu of the Mercury tool. The main menu is shown by Figure 8. In order to activate the functionality regarding each menu item, the user must click on the menu item by using the left mouse button. After that, and depending on the chosen menu item, a window may appear in order to show other options to be selected by the user. In addition, some menu items also have a keyboard shortcut. Figure 9 shows the items presented in the **File** menu and, as we can see, some have keyboard shortcuts associated with them.

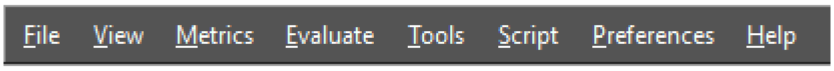


Figure 8: Main Menu

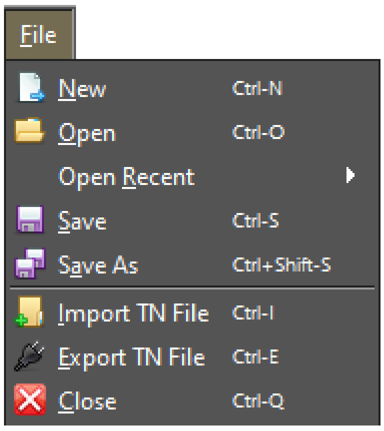


Figure 9: Menu File

Next we will explain the menu items present in the File menu.

New. Creates a new project, in which all the modeling views are initialized empty, except for the RBD and FT views, where each one starts with a default component. Shortcut: Ctrl + N.

Open. Opens a project already created in Mercury. The tool only allows opening files with the ".xml" extension. In order to open files created by considering the ".TN" format, the user must go to the "Import TN File" item. Shortcut: Ctrl + O.

Open Recent. It shows a list of the twenty most recent projects created or opened on Mercury.

Save. Saves the current project. For the first time that a project is saved, a window is displayed in order for the user to choose the location and name of the file to be created. Shortcut: Ctrl + S.

Save As. Saves the current project, defining another file name and/or location for it. Shortcut: Ctrl + Shift + S.

Import TN File. Imports files created in the ".TN" standard. Shortcut: Ctrl + I.

Export TN File. Exports the current project considering the ".TN" standard. Shortcut: Ctrl + E.

Close. Closes the tool. Shortcut: Ctrl + Q.

Figure 10 shows the menu **View**. In the menu **View**, the user can show or hide the six views provided by the tool: RBD, FT, EFM, SPN, CTMC, and DTMC. Figures 11 and 12 depict the main window with the six views visible and hidden, respectively.

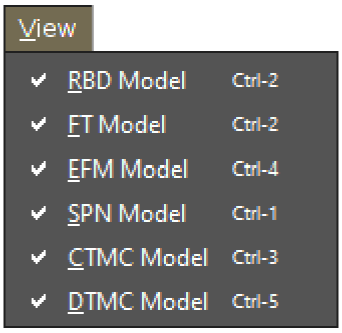


Figure 10: Menu View

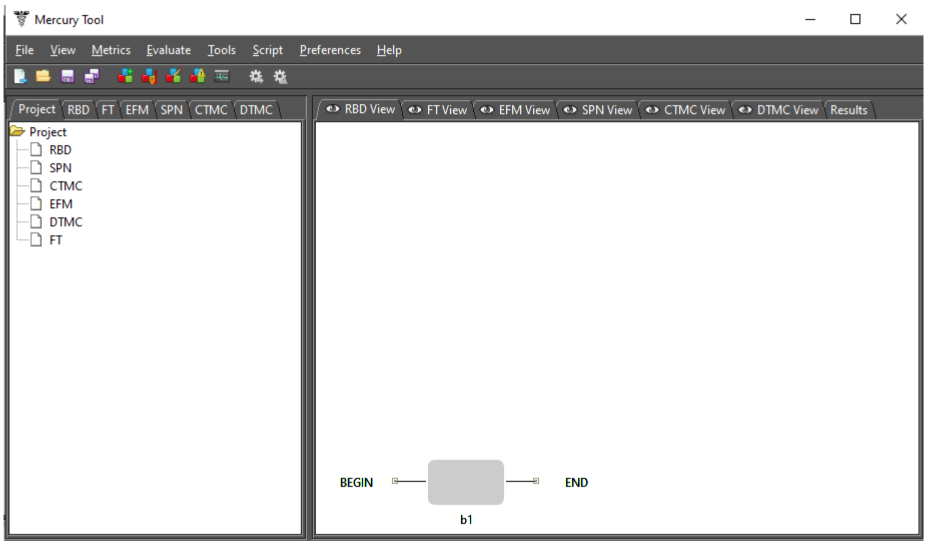

Mercury Tool
File View Metrics Evaluate Tools Script Preferences Help
Project RBD FT EFM SPN CTMC DTMC
RBD View FT View EFM View SPN View CTMC View DTMC View Results
Project
RBD
SPN
CTMC
EFM
DTMC
FT
BEGIN
END
b1

Figure 11: Six Views Visible

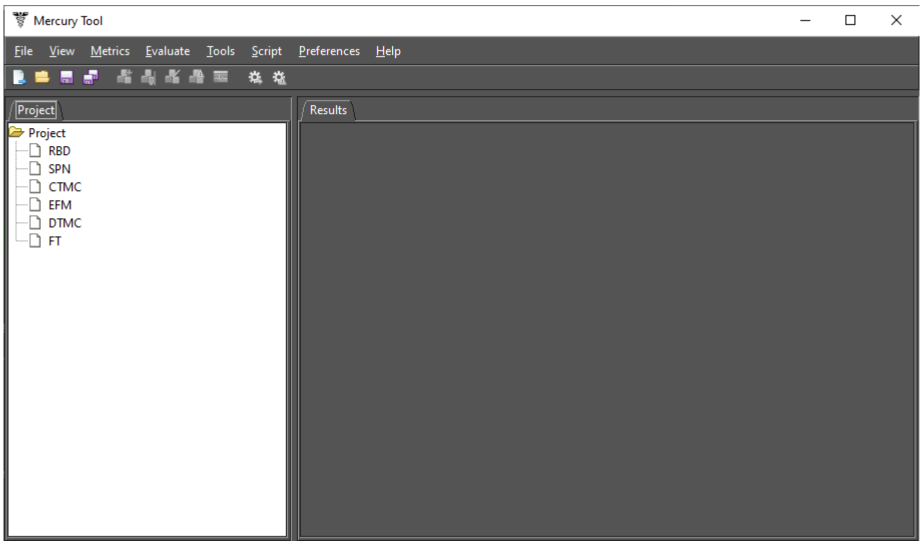


Figure 12: Six Views Hidden

Figure 13 shows the menu **Evaluate** which has a menu group for each formalism supported by the Mercury tool. A menu group is enabled only when the corresponding model view is selected in the drawing area. The menu items presented in each menu group for each formalism are presented in other sections ahead.

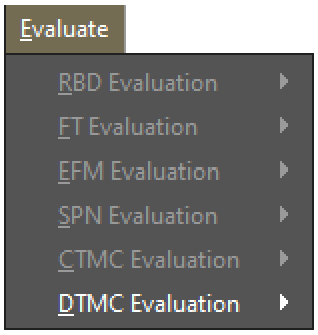


Figure 13: Menu Evaluate

Figure 14 shows the menu **Tools** and its features are described as follows:

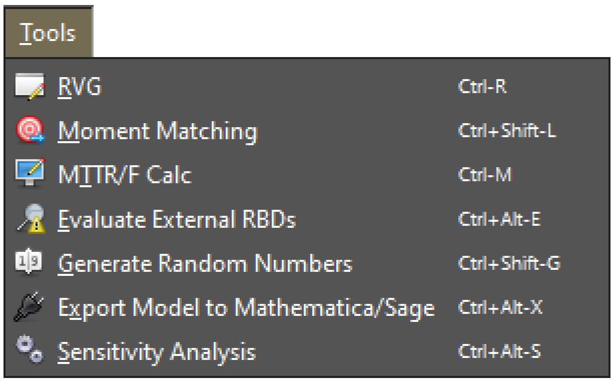

Tools
RVG Ctrl-R
Moment Matching Ctrl+Shift-L
MTTR/F Calc Ctrl-M
Evaluate External RBDs Ctrl+Alt-E
Generate Random Numbers Ctrl+Shift-G
Export Model to Mathematica/Sage Ctrl+Alt-X
Sensitivity Analysis Ctrl+Alt-S

Figure 14: Menu Tools

RVG stands for **Random Variate Generator**. It is an additional tool responsible for generating random numbers, which includes several probability distributions. Statistical summaries are also provided for the generated numbers (e.g., standard deviation, variance, mean, skewness, and kurtosis) and the data can be exported to files for use in external applications. Moreover, the generator is used by the SPN simulator, in which the user can choose different probability distributions associated with timed transitions.

Moment Matching [2]. It is another auxiliary tool available in the Mercury environment. By adopting this tool, users can estimate which exponential-based probability distribution best fits the mean (first moment) and standard deviation (second moment) of user data for a given model parameter.

MTTR/F Calculator. It allows one to compute the MTTR (Mean Time To Repair) and MTTF (Mean Time To Failure) of the system in analysis by using availability and reliability curve as input parameters. The reliability is a time-dependent metric that gives the probability that an item will perform a required function, under stated conditions, for a stated period of time.

The system reliability $R(t)$ can be defined as follows [3]:

$$R(t) = exp\Big[\int_0^t \lambda(t)dt\Big]$$

where $\lambda(t)$ corresponds to the failure rate over the time t. However, if $\lambda(t)$ is constant, the reliability can be evaluated as,

$$R(t) = e^{-\lambda t}.$$

For instance, suppose the system failure rate is $0.5[h^{-1}]$, then the reliability curve should be the following (see Figure 15):

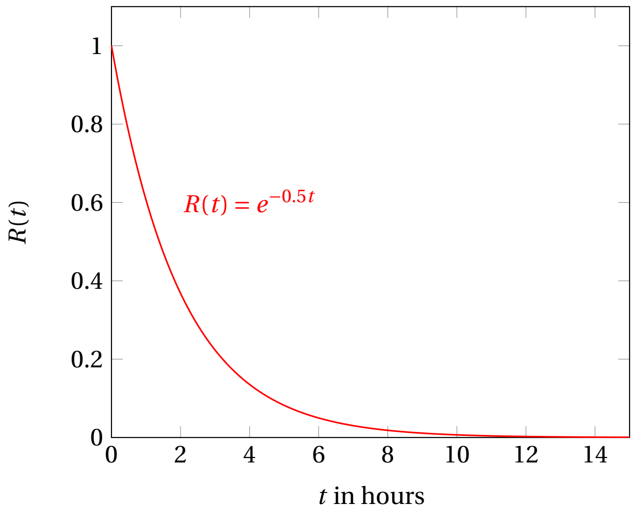


Figure 15: Reliability Over Time

The system MTTF and MTTR can be calculated using the following expressions [3]:

$$MTTF = \int_0^\infty R(t)dt$$

$$MTTR = \frac{MTTF}{availability} - MTTF$$

Figure 16 shows the MTTR/F Calculator window, in which users should specify the *Availability* and a CSV file with no headers and two columns specifying the evaluation time and reliability value. Button *File* allows users to point to the file containing time and reliability values.

After specifying the parameters, the user should compute the results by pressing the button *Calculate*. Figure 17 shows an example of the result obtained after computing MTTR and MTTF values. For a more detailed example, please refer to the following video `https://youtu.be/Hmu5DX3CJCg`.

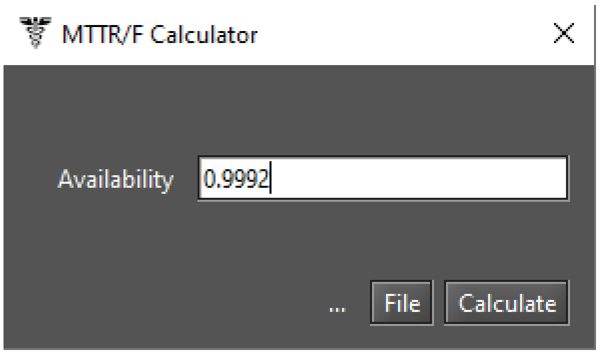
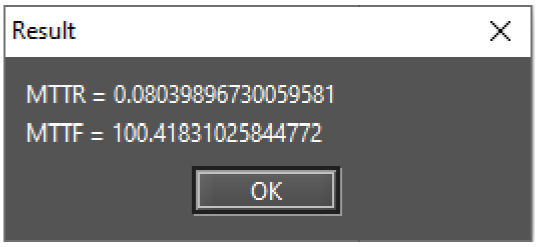


Figure 16: MTTR/F Calculator



MTTR = 0.08039896730059581
MTTF = 100.41831025844772

Figure 17: MTTR/F Calculator Results

Evaluate External RBDs. It is a tool for computing availability metrics from external RBD files, created following a specific format.

Generate Random Numbers. It is a tool for generating random numbers that follow a probability distribution. It supports a large number of probability distributions. Firstly, the user needs to enter the size of the sample and select the probability distribution that will guide the generation of numbers. After that, it is necessary to define the value for each parameter of the chosen distribution. Before starting the process, the user must choose the location where the file containing the sample will be saved.

Export Model to Mathematica/Sage. It is an option for saving CTMC models in Wolfram Mathematica language/SageMath format.

Sensitivity Analysis. It allows performing sensitivity analysis on SPN models. The user must select between two sensitivity analysis methods: "Design of Experiments" or "Sensitivity Indices". The former uses the standard method of analysis of variance to identify the effect of each factor on model results. The latter uses the technique of percentage difference, so it requires a minimum and maximum value for each parameter in order to compute the corresponding percentage variation on the chosen metric. Mercury is able to compute the sensitivity of an SPN measure with respect to each SPN delay parameter as well as with respect to the parameters of CTMC sub-models that provide the value for any SPN transition.

Figure 18 shows the menu **Script**. In the menu **Script**, the user can generate the script representing the model on the active view or create a script from scratch. By clicking on the "Generate script" menu, the script representing the active model is generated and can be evaluated and modified in the Script Editor. Figure 19 shows the script editor with a script representing an SPN model. Also, Mercury offers scripts as examples in this menu. Just click on the menu item representing the desired script and the "Script Editor" will open it.

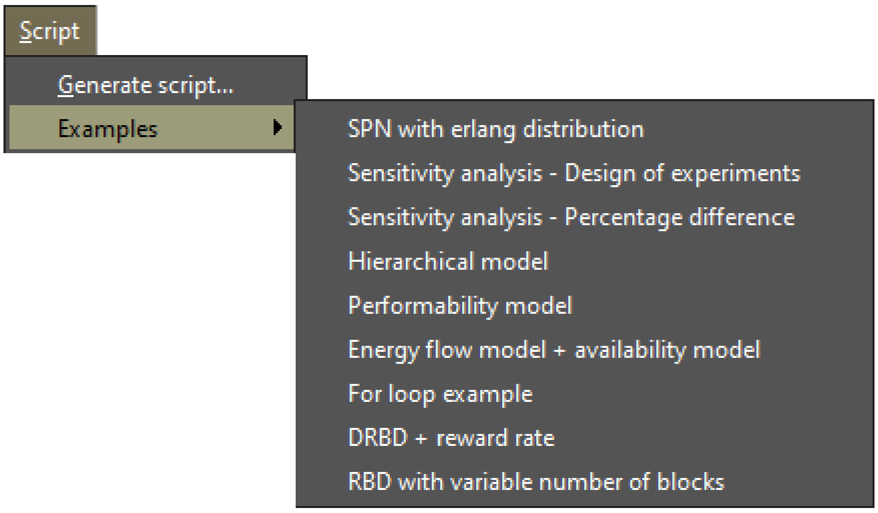
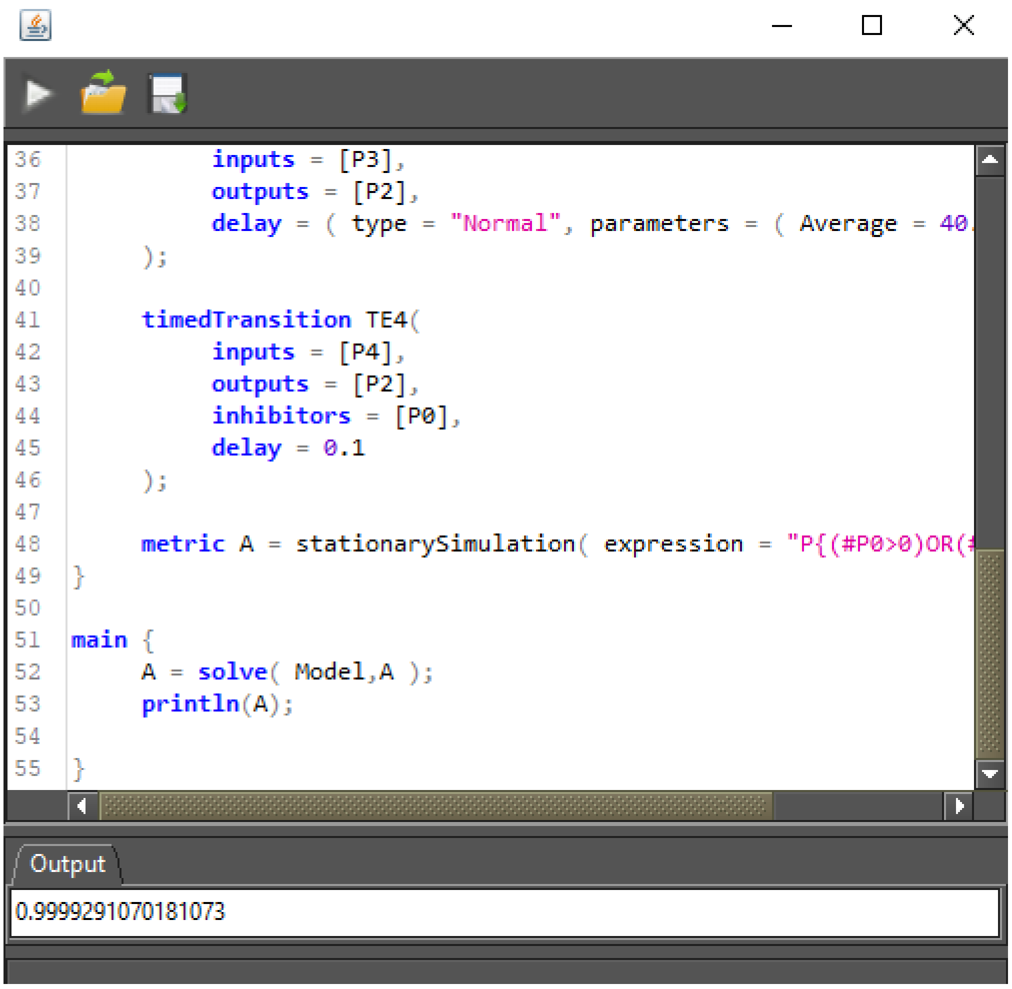

Script
Generate script...
Examples
SPN with erlang distribution
Sensitivity analysis - Design of experiments
Sensitivity analysis - Percentage difference
Hierarchical model
Performability model
Energy flow model + availability model
For loop example
DRBD + reward rate
RBD with variable number of blocks

Figure 18: Menu Script



```
          inputs = [P3],
          outputs = [P2],
          delay = ( type = "Normal", parameters = ( Average = 40
     );

     timedTransition TE4(
          inputs = [P4],
          outputs = [P2],
          inhibitors = [P0],
          delay = 0.1
     );

     metric A = stationarySimulation( expression = "P{(#P0>0)OR(
}

main {
     A = solve( Model,A );
     println(A);

}
```

```
0.9999291070181073
```

Figure 19: Script Editor

1.4 Main Toolbar

The main toolbar provides access to some of the most used features in the Mercury tool. The toolbar is present at the top of the Mercury tool window, just below the menu bar. Figure 20 shows the command buttons on this toolbar, in which each button is represented as an icon. These buttons are important and often used to create new projects or save the current project, for example. The following items describe the functionality of each command button.

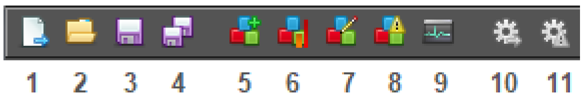


Figure 20: Main Toolbar

1. **New.** Creates a new project. Shortcut: Ctrl + N.

2. **Open.** Opens a project. Shortcut: Ctrl + O.

3. **Save.** Saves changes to the current file. Shortcut: Ctrl + S.

4. **Save As.** Saves the current project with a new name, which can be entered in a subsequent dialog. Shortcut: Ctrl + Shift + S.

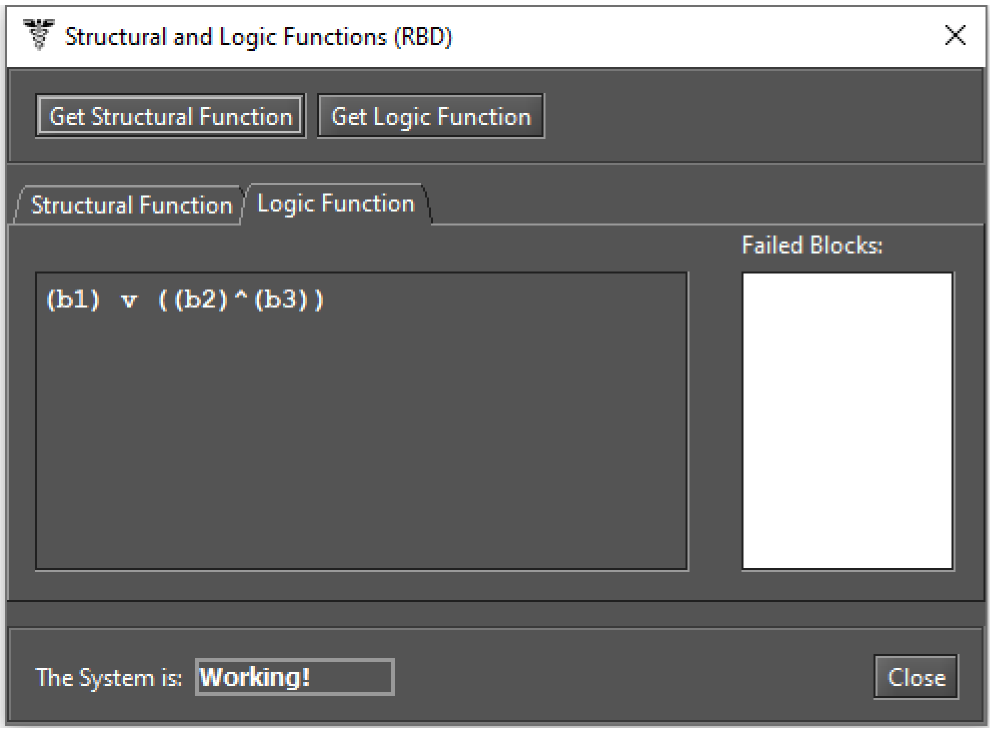
5. **RBD Evaluation.** It makes it possible to carry out a myriad of evaluations in the RBD model. It is enabled only when the RBD view is active.

6. **RBD Bounds Evaluation.** This function performs the bounds evaluation in the RBD model. It is enabled only when the RBD view is active.

7. **RBD Experiment.** It allows the user to perform an experiment on the RBD model. It is enabled only when the RBD view is active. Shortcut: Ctrl + F11.

8. **RBD Importance Measures.** This function calculates the importance of each component of the RBD model. It is enabled only when the RBD view is active. Shortcut: Ctrl + F4.

9. **Structural and Logic Functions (RBD).** It obtains the functions of the RBD model. It is enabled only when the RBD view is active. The " Structural Function " is a function related to the states of the components. The system and each component may be in a working or failed state. The system state is a binary random variable that is determined by the component's states. If the states of the components are known, the state of the system is also known. The state can be changed by accessing the properties of the blocks and when the state of the block is "failed", the component is indicated by a broken icon. To obtain the functions, the user can click on this button and choose an option to show between logic or structural. Figure 21 shows an example of a logical function for a system without failed blocks.



```
(b1) v ((b2)^(b3))
```

Working!

Figure 21: Logic Function (RBD)

10. **SPN Stationary Simulation.** This button starts the SPN stationary simulator. It is only enabled when the SPN View is the current view.

11. **SPN Transient Simulation.** This button starts the SPN transient simulator. It is only enabled when the SPN View is activated. Only single metrics related to Probability and Expectation are accepted when transient simulations are adopted. For instance, metrics like P{#P0=1} and E{#P1} are accepted. However, metrics like (P{#P0=1}*2)/E{#P3} and E{#P1}*3 are not allowed in transient simulations.

1.5 Drawing Area

The Mercury tool has six views for modeling (RBD, FT, EFM, SPN, CTMC, and DTMC) and also another one (Results) to display the results generated by the simulators. The user must click on the desired view tab in order to enable its visualization. Moreover, the views can be enabled or disabled. To do this, the user must check or uncheck the view in the View menu. A drawing area is available for each formalism supported by Mercury.

The drawing area is a blank area where the user can add components. Figure 22 shows the drawing area of the SPN view. The components available for insertion, depending on the formalism adopted by the user. In general, to add a component, it is necessary to click on the button that represents the component on the toolbar and right after click inside the drawing area.

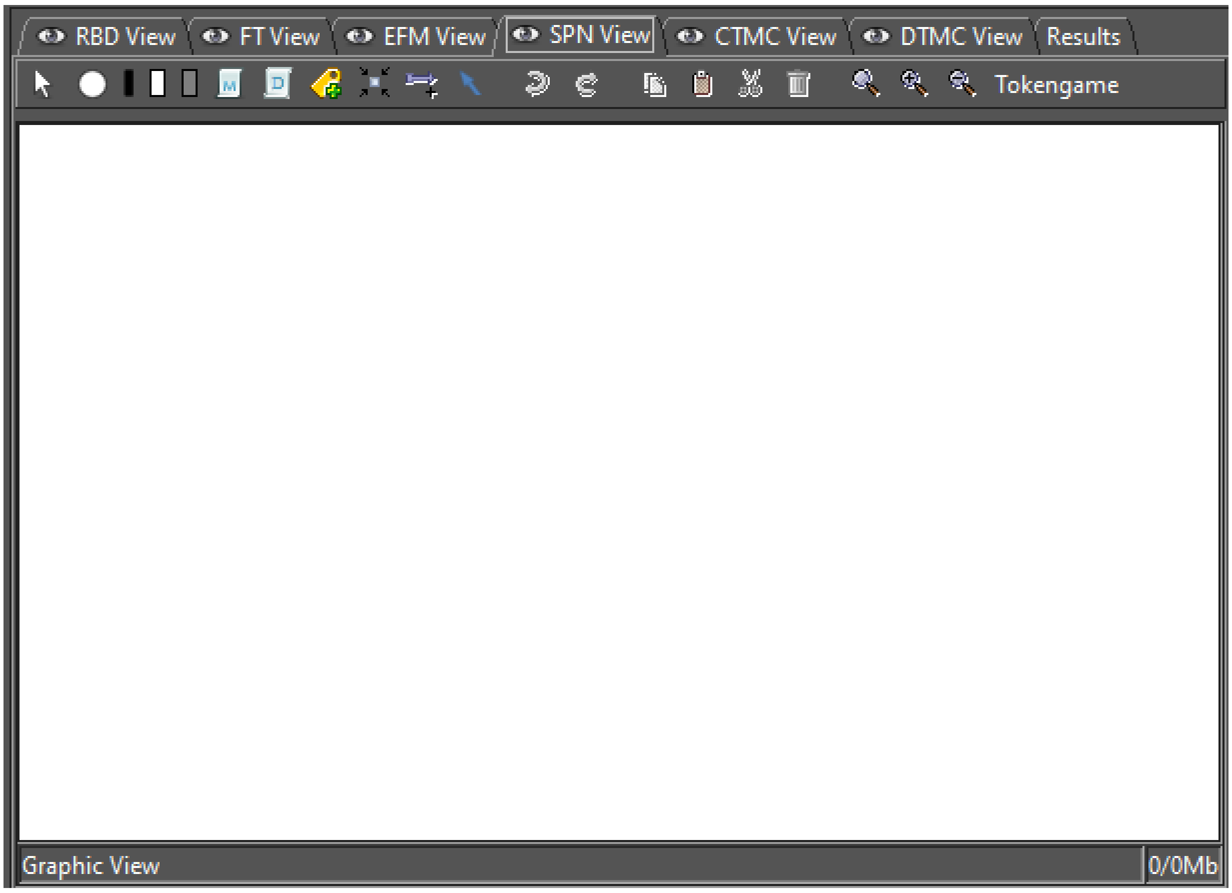


Figure 22: Drawing Area

In addition, on the left side of the tool, there is a tab for each formalism supported by Mercury — RBD, FT, EFM, SPN, CTMC, and DTMC — as we can see in Figure 23, in which all components available in the current project can be accessed. In this example, the EFM tab is active and three components are being displayed. Each time a component is inserted to the drawing area, the panel corresponding to the formalism on the left side of the main window is updated.

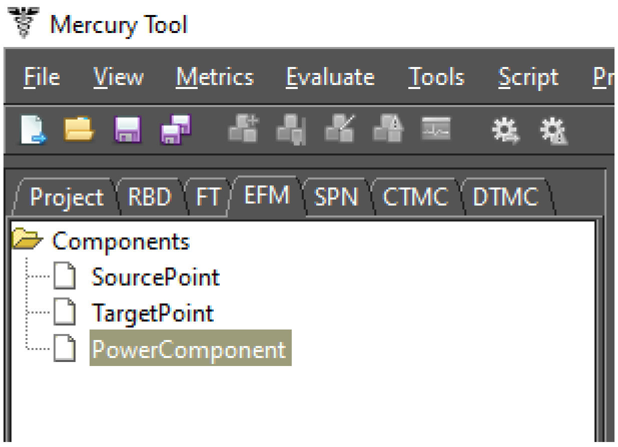


Figure 23: Left-Side Panel

2 SPN Modeling and Evaluation

Regarding the SPN view, users can create SPNs models by adding components such as places and immediate and stochastic transitions. Mercury is a complete tool for modeling SPNs. Figure 24 shows an SPN model in which there are a timed transition and an immediate transition as well as two places. Below we detail the process for modeling SPNs by using the Mercury tool.

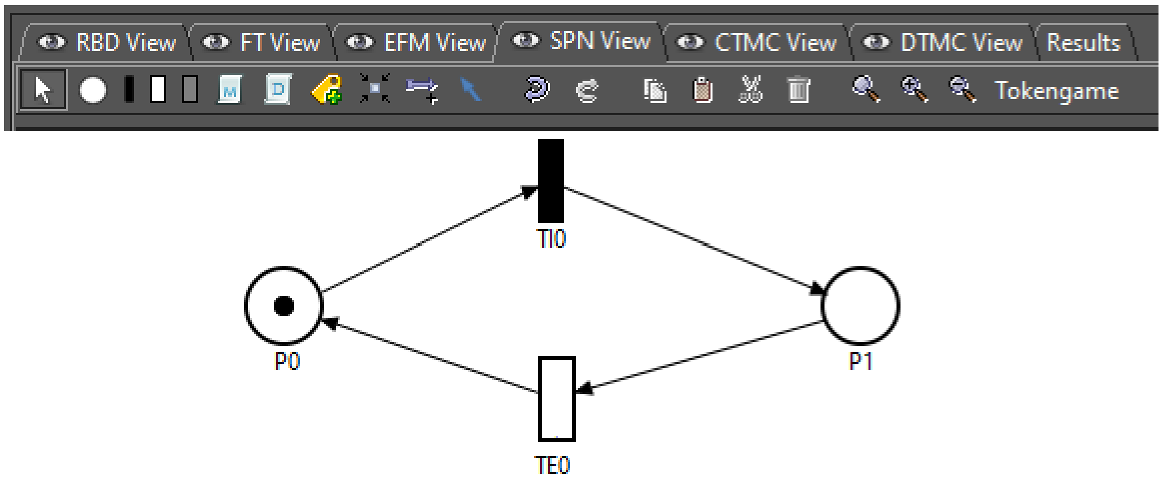


Figure 24: SPN Model

The buttons on the SPN toolbar are the ones used to model SPNs. The toolbar is visible when the SPN view is active. Below we provide descriptions related to each of the buttons on this toolbar (see Figure 25).

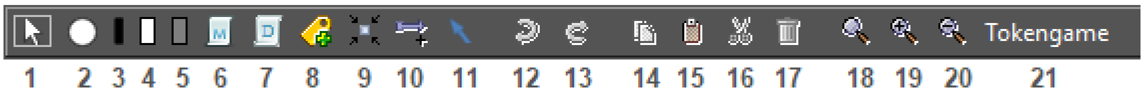


Figure 25: SPN Toolbar

1. **Selection Mode.** Activates the selection mode. This mode makes it possible to select model components in the drawing area.

2. **Add Place.** Adds places to the model. To do this, the user should click on the Add Place button and then on the desired location on the drawing area.

3. **Add Immediate Transition.** Adds immediate transitions to the model. To do this, the user should click on the Add Immediate Transition button and then on the desired location on the drawing area.

4. **Add Exponential Transition.** Adds exponential transitions to the model. To do this, the user should click on the Add Exponential Transition button and then on the desired location on the drawing area.

5. **Add Non-Exponential Transition.** Adds non-exponential transitions to the model. To do this, the user should click on the Add Non-Exponential Transition button and then on the desired location on the drawing area.

6. **Add Reward Measure.** Adds reward measures to the model. To do this, the user should click on the Add Reward Measure button and then on the desired location on the drawing area.
7. **Add Definition.** Adds variables to the model. To do this, the user should click on the Add Definition button and then on the desired location on the drawing area.
8. **Add Label.** Adds labels to the model. To do this, the user should click on the Add Label button and then on the desired location on the drawing area.
9. **Show/Hide Arcs Labels.** Shows and hides the labels of the arcs.
10. **Connection Mode.** Enables or disables the connection mode. This connection mode when enabled allows the user to connect components in the drawing area by using arcs. Connecting places to transitions and vice versa.
11. **Default/Inhibitor Arc.** It allows the user to choose the type of arc that will be used to connect places to transitions. The user has the possibility to use two types of arcs: standard arc and inhibitor arc. When a new project is created, the standard arc is active by default.
12. **Undo.** Undo recent actions in the drawing area. All recent changes are stored and can be rolled back, one after one. Shortcut: `Ctrl+Z`.
13. **Redo.** Redo recent undone actions in the drawing area. Shortcut: `Ctrl+Y`.
14. **Copy.** Copy the selected components in the drawing area to the clipboard. Shortcut: `Ctrl+C`.
15. **Paste.** Insert components that are in the clipboard into the drawing area. Shortcut: `Ctrl+V`.
16. **Cut.** Remove the selected components from the drawing area and places them in the clipboard. Shortcut: `Ctrl+X`.
17. **Delete.** Removes selected components from the model.
18. **Default Scale.** Apply the standard scale to the drawing area.
19. **Scale Up.** Each click scales up the drawing image by 10% percent (zoom in).
20. **Scale Down.** Each click scales down the drawing image by 10% percent (zoom out).
21. **Token Game.** Token Game is a function that allows us to graphically evaluate the behavior of the SPN model being developed. When activating the Token Game, transitions enabled for the current marking are highlighted, and the user has the option to double-click on any of them in order to fire it. By firing, a new marking is reached and, depending on it, new transitions are enabled and others become disabled. In this way, the user can visually test whether the model is adequately representing the behavior of what is being modeled.

When right-clicking on an SPN component, a pop-up menu is displayed. All SPNs components have a popup menu that provides at least two menu items:

- **Remove.** Remove the selected component from the model.
- **Properties.** Show the "Properties" dialog box that allows the user to change the properties of the selected component.

Figure 26 shows as an example a pop-up menu displayed when right-clicking on a transition.

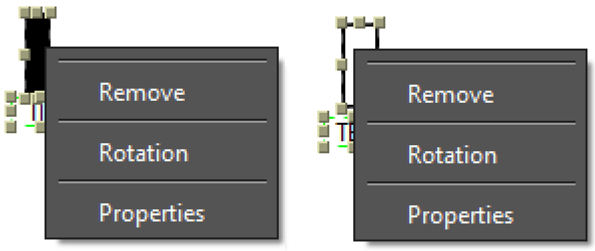


Figure 26: Transition Menu

The menu displays three menu items. Rotation is a menu item specific to transitions.

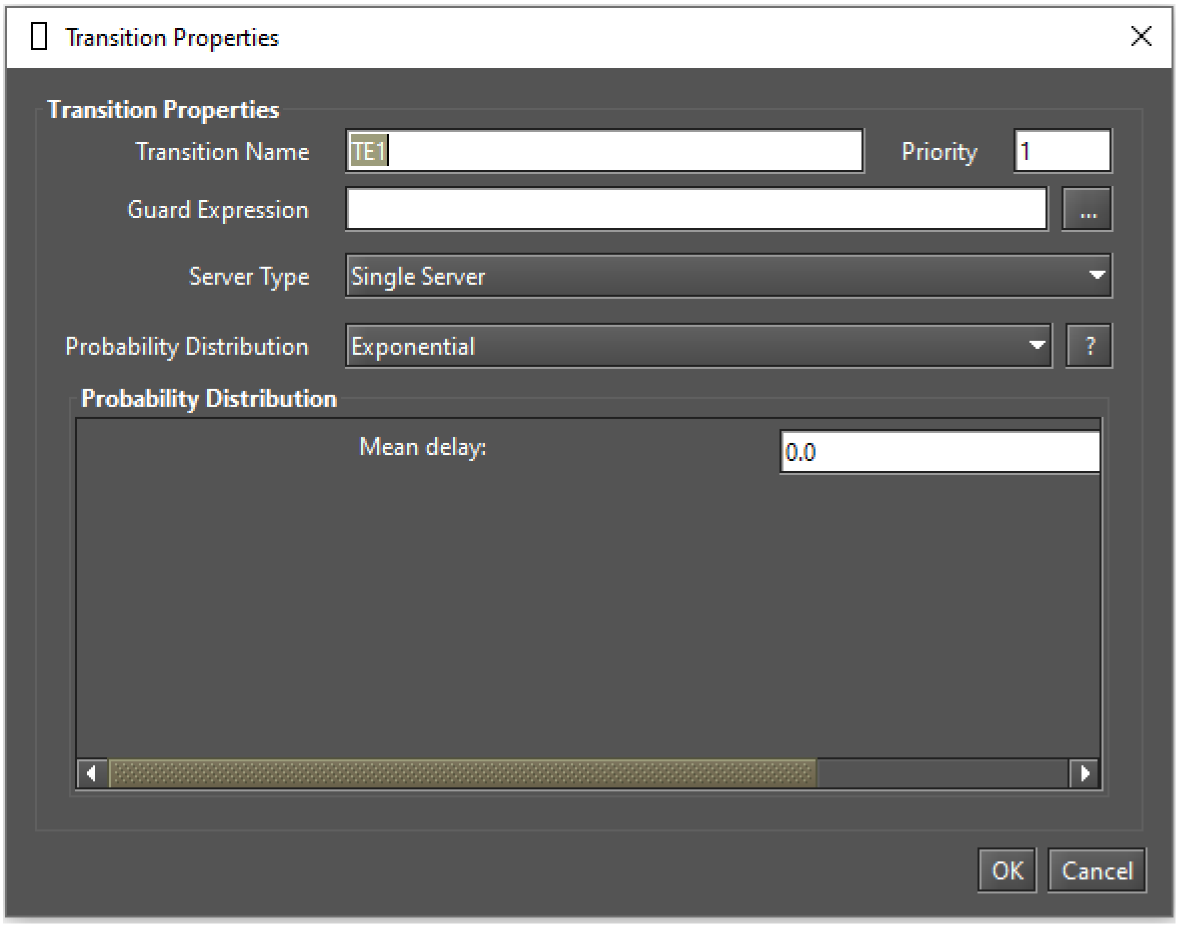
- **Rotation.** Rotate the selected transition. A transition can be positioned horizontally or vertically. Figure 27 shows an immediate transition positioned horizontally.

Figure 27: Immediate Transition Positioned Horizontally

Now let us see the properties of a timed transition. To view the properties of the component, the user must press the right mouse button on the desired component, as shown in the Figure 26, and then click on the item Properties. Figure 28 shows the properties of a timed transition.

The properties are respectively:

- **Transition Name.** Name of the transition.
- **Priority.** Firing priority assigned to the transition. The higher the priority, the higher the precedence to fire. Immediate transitions always take precedence over timed transitions.
- **Guard Expression.** It is a boolean expression that allows a transition to be enabled and can be fired. Besides the fact that the current marking makes this possible, the transition only becomes enabled and can be fired when the guard expression assigned to it is evaluated true.
- **Server Type.** Corresponds to the firing semantics associated with that timed transition. The user can select one of the only two options available. They are the single server semantic (SSS) and infinite server


Transition Properties

Transition Properties

Transition Name: TE1

Priority: 1

Guard Expression:

Server Type: Single Server

Probability Distribution: Exponential

Probability Distribution

Mean delay: 0.0

Figure 28: Timed Transition Properties

semantic (ISS). In the SSS, the transition only becomes enabled and can fires only once every instant. In the ISS the number of tokens in the input places of that transition defines its enabling degree. Thus, the enabling degree defines the degree of parallelism for that transition.

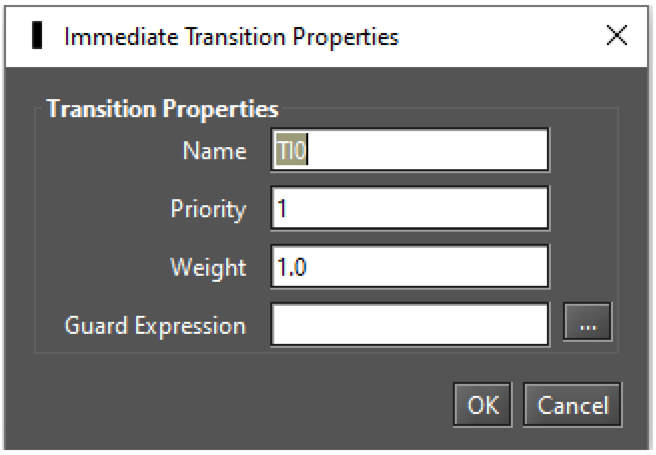
- **Probability Distribution:** The Mercury tool supports a large number of probability distributions. When all timed transitions composing the model are exponential, this model may be evaluated by carrying numerical analysis out as well as by simulation. On the other hand, when there is at least one non-exponential timed transition, the model only can be evaluated by simulation. Moreover, depending on the chosen distribution, fields related to the parameters of that distribution may appear in order for the user to enter their values. The tool supports the following probability distributions:
  - Beta
  - Binomial
  - Burr
  - Cauchy
  - Chi-squared
  - Deterministic

  - Discrete Uniform
  - Erlang
  - Exponential
  - F Fisher–Snedecor
  - Frechet
  - Gamma
  - Generalized Extreme Value
  - Generalized Pareto
  - Geometric
  - Hypergeometric
  - Logistic
  - Log-logistic
  - Log-normal
  - Nakagami
  - Normal
  - Pareto
  - Pearson Type 6
  - Poisson
  - Rayleigh
  - Student’s t-distribution
  - Triangular
  - Uniform
  - Weibull

  Some probability distributions require a delay parameter that corresponds to the delay to fire the transition. In addition to the delay, other parameters may be required, depending on the chosen distribution. For example, exponential distribution requires only one parameter: mean delay. On the other hand, Erlang distribution requires two parameters: mean delay and shape. Normal distribution requires two parameters: mean and standard deviation. Each probability distribution has its parameters.

Now let us see the properties of an immediate transition. Figure 29 shows the properties of a immediate transition.

The properties are respectively:


Immediate Transition Properties

Transition Properties

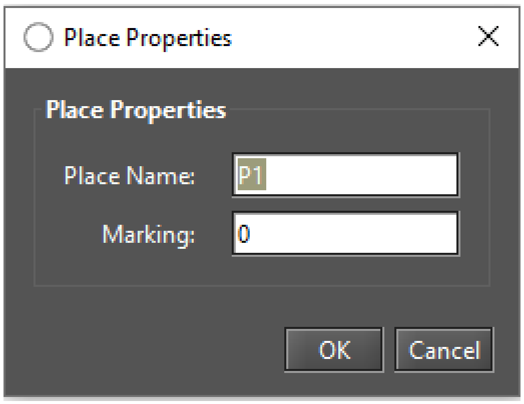
Name	TI0
Priority	1
Weight	1.0
Guard Expression	

Figure 29: Immediate Transition Properties

- **Name.** Name of the transition.
- **Priority.** Firing priority assigned to the transition. The higher the priority, the higher the precedence to fire. Immediate transitions always take precedence over timed transitions. Even in the case when an immediate transition and an exponential transition are enabled at the same marking, and the immediate transition has a lower priority than the exponential transition, the former fires first.
- **Weight.** Weight of the transition.
- **Guard Expression.** It is a boolean expression that allows a transition to be enabled and can be fired. Besides the fact that the current marking makes this possible, the transition only becomes enabled and can be fired when the guard expression assigned to it is evaluated true.

Now let us see the properties of an SPN place. Figure 30 shows the properties of a place. The properties are respectively:

- **Name.** Name of the place.
- **Marking.** The number of tokens in that place.


Place Properties

Place Properties

Place Name:	P1
Marking:	0

Figure 30: Place Properties

Figure 31 shows as an example a pop-up menu displayed when right-clicking on an arc.

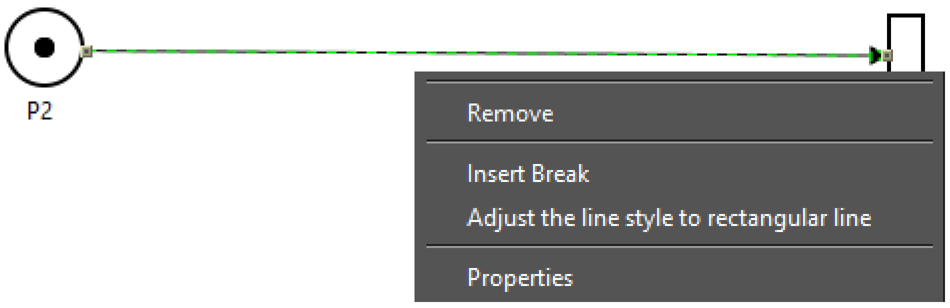


Figure 31: Arc Menu

The menu displays four menu items. “Insert break” and “Adjust the line style to...” are menu items specifics to arcs.

- **Insert Break.** Insert a break point at the clicked location, and through that break point it is possible to move that point to the desired location, changing the shape of the arc.

- **Adjust the line style to # line.** Mercury supports two line styles: rectangular and curved. The curved line format is the default style. To change to the rectangular style just click on the corresponding menu item. Figure 32 shows an SPN with a rectangular arc and Figure 33 shows an SPN with a curved arc.

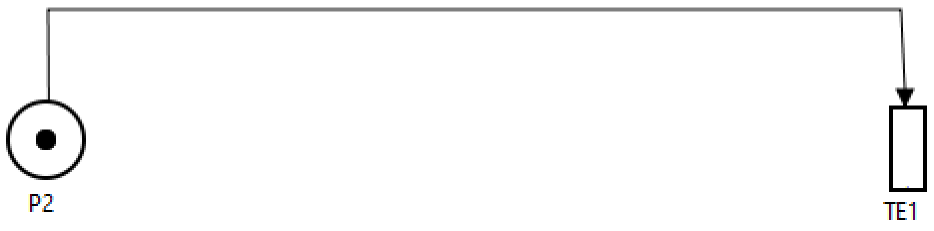


Figure 32: SPN with a Rectangular Arc

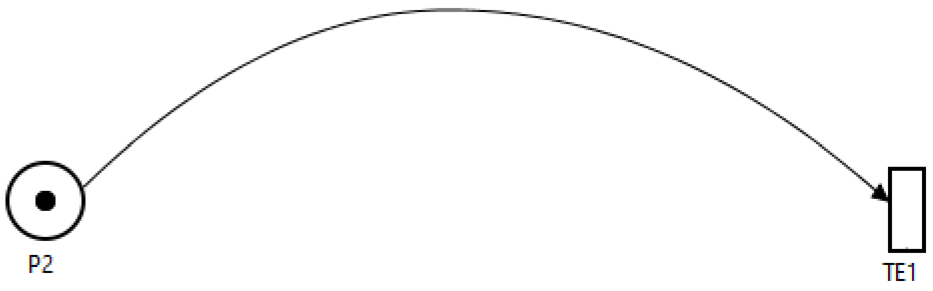
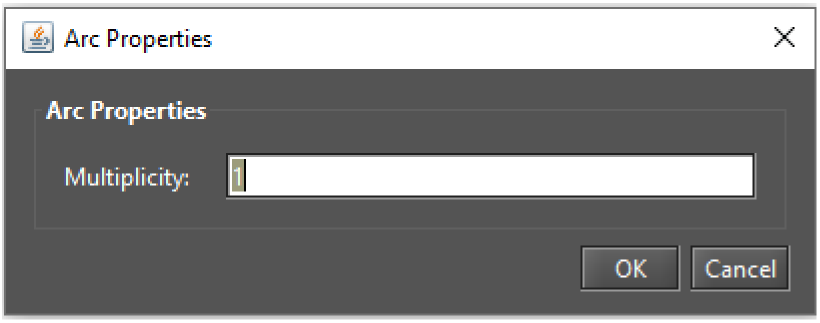


Figure 33: SPN with a Curved Arc

Figure 34 shows the properties of an arc. SPN arcs have only one property:

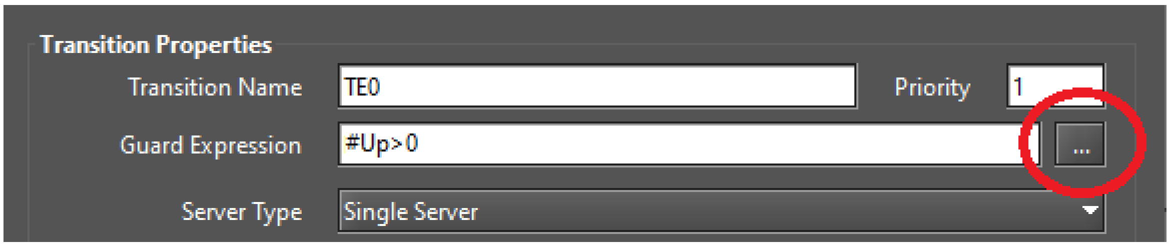
- **Multiplicity:** The multiplicity of the arc, that is, the weight of that arc. The number of tokens required when the arc is an output arc of a place or the number of tokens created in a place when the arc is an input arc for that place.



Multiplicity: 1

Figure 34: Arc Properties

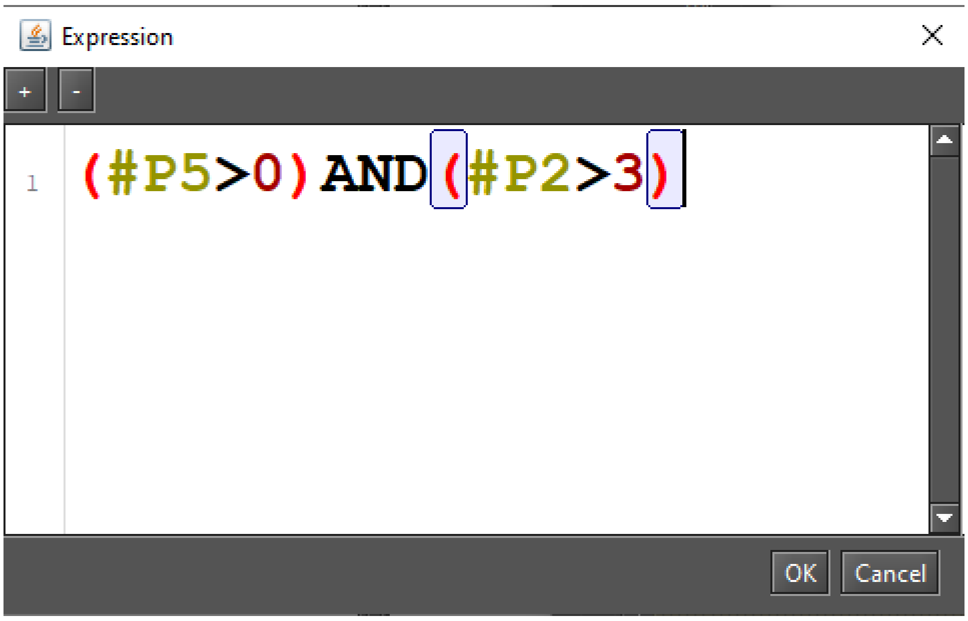
In some dialog boxes, it is possible to see the presence of a button with an ellipsis as a description. This button is always next to a text field (see Figure 35).



Transition Name: TE0
Priority: 1
Guard Expression: #Up>0
Server Type: Single Server

Figure 35: Accessing the Expression Editor

By clicking on this button opens the Expression Editor (see Figure 36).



```
(#P5>0) AND (#P2>3)
```

Figure 36: Expression Editor

Expression Editor is a text editor that makes it easy to create expressions for defining guard expressions and metrics. It is a simple editor that highlights parentheses, brackets, and braces as well as some keywords. In addition, it has a button that reduces the font of the text and another one that increases it, thus facilitating the process of defining large and complex expressions.

The Mercury tool offers for the evaluation of SPN models, the stationary and transient simulations as well as stationary and transient analyses. These features are available in the Evaluate menu, in the SPN Evaluation option as well as in the command buttons on the main toolbar (see Figure 37 and 38). Next, we will present the stationary and transient simulators and, afterward, the SPN analyses.

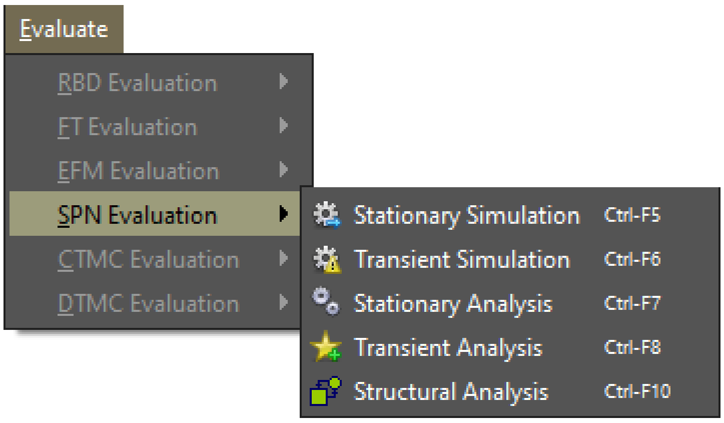

Evaluate
RBD Evaluation
FT Evaluation
EFM Evaluation
SPN Evaluation
CTMC Evaluation
DTMC Evaluation
Stationary Simulation Ctrl-F5
Transient Simulation Ctrl-F6
Stationary Analysis Ctrl-F7
Transient Analysis Ctrl-F8
Structural Analysis Ctrl-F10

Figure 37: SPN Menu

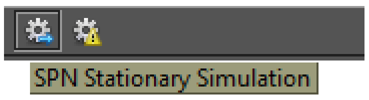


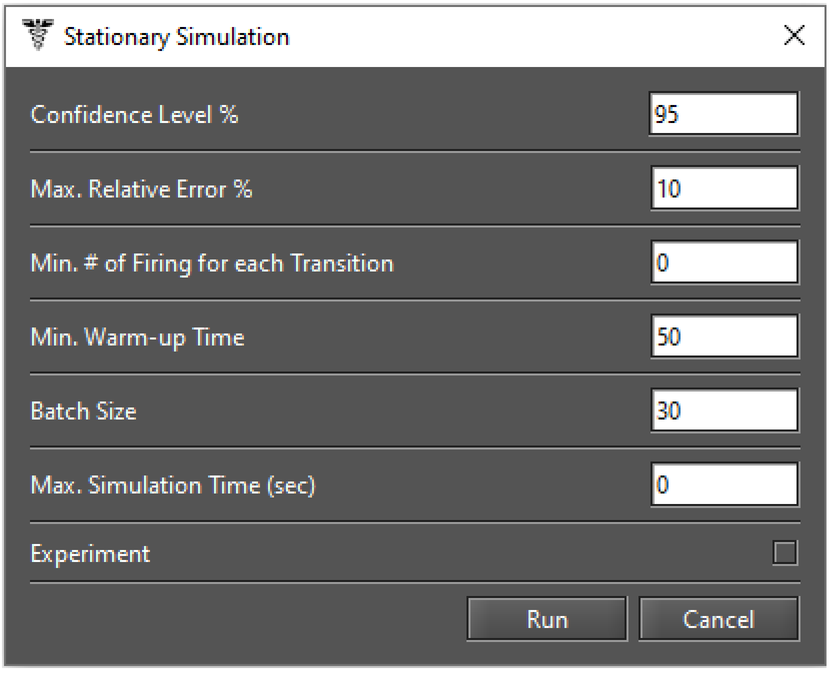
Figure 38: Simulation Buttons

2.1 SPN Simulation

Models that have non-exponential transitions can only be evaluated through simulation. The Mercury tool has two types of simulators for SPN models. Mercury has a stationary simulator to obtain stationary metrics and a transient simulator to obtain time-dependent metrics. Next, we will present the stationary simulator and, in the next subsection, we will present the transient simulator.

2.1.1 Stationary Simulation

Figure 39 shows the input parameters for the stationary simulator. These parameters are detailed below:


Stationary Simulation

Confidence Level %	95
Max. Relative Error %	10
Min. # of Firing for each Transition	0
Min. Warm-up Time	50
Batch Size	30
Max. Simulation Time (sec)	0
Experiment	☐

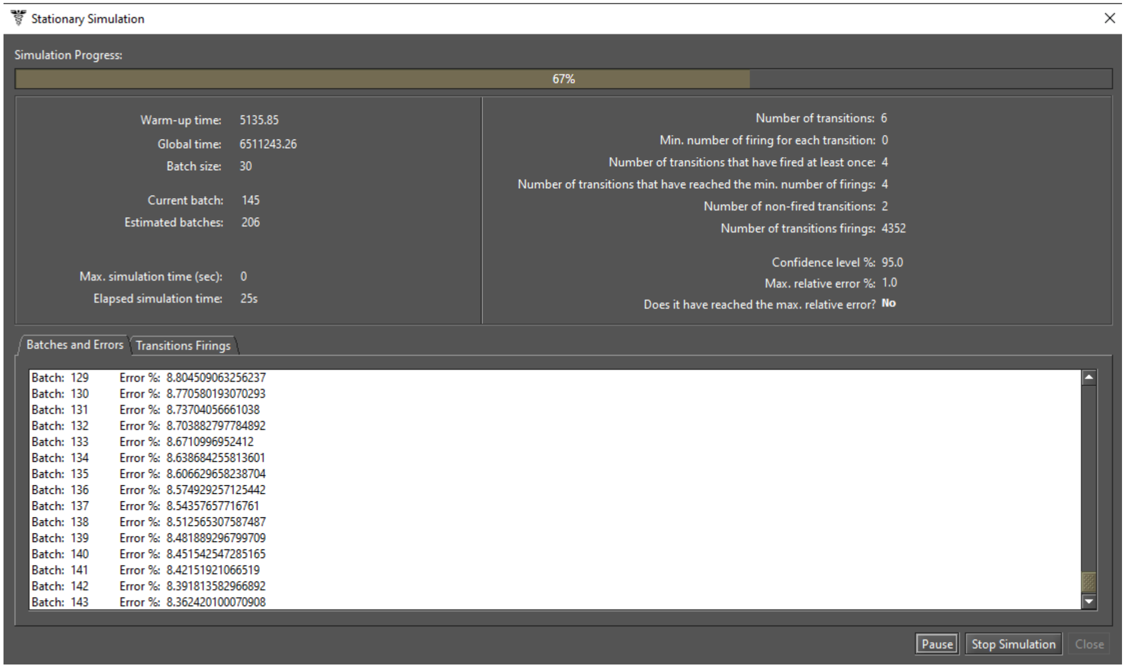
Figure 39: Stationary Simulation

- **Confidence Level.** The confidence interval for obtaining the metrics.
- **Max. Relative Error %.** Defines the maximum relative error that is one of the stop conditions of the simulation.
- **Min. of firing for each Transition.** Sets the minimum number of firings for each transition. This number of firings is another condition to stop the simulation. When set to 0, the simulator does not consider the number of firings in order to stop the simulation. Entering a value greater than 0, the simulation stops when the number of firings for each transition is equal to the defined value.
- **Min. Warm-up Time.** Sets the minimum warm-up period. The warm-up phase is the period when the model is not considered to be in a steady-state, and the metrics are not collected in that period. There are some methods for estimating whether the model has entered a stationary phase, but Mercury requires the user to define a value for the warm-up time. In future versions, we plan to implement and make available

to users some estimation methods for the warm-up phase. As we are evaluating stochastic models, it is expected that the warm-up time is not an exact value for each simulation performed. Therefore, the user defines a minimum warm-up time and, once the global simulation time is equal to or greater than the warm-up time defined by the user, the simulation begins to collect the metrics, generate the batches, and calculate statistics.

- **Batch Size.** Defines the number of samples that will constitute each batch in the simulation.
- **Max. Simulation Time (sec).** It is used to define the maximum simulation time. This time corresponds to the physical time and must be defined in seconds. If one of the stop conditions is not met before this maximum time is reached (maximum relative error and number of firings for each transition), then the simulation stops when this time is reached.
- **Experiment.** Experiment allows simulation by changing the value of a given variable considering a step size and a minimum and maximum value for that variable. For each change in the value of the variable, a new simulation is performed. At the end of an experiment, the Mercury tool plots a graph showing the impact of the variable's value variation on the evaluated metric, showing the average values with their confidence intervals.

Figure 40 shows the stationary simulator in action.

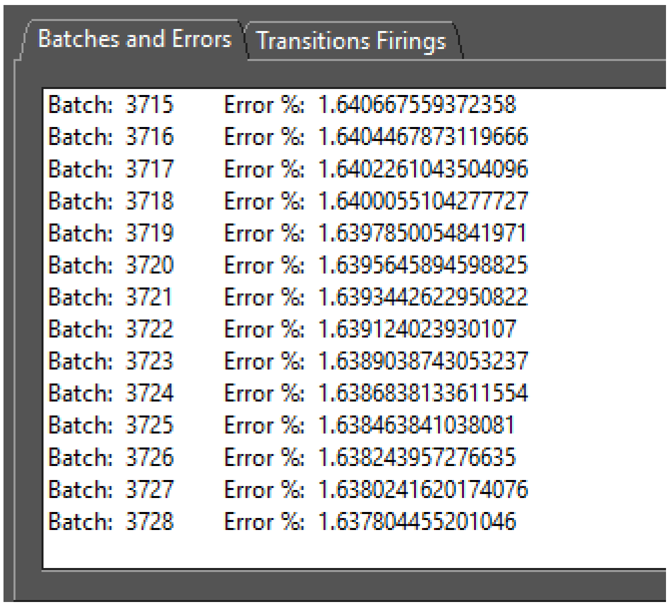


```
Warm-up time:  5135.85
Global time:  6511243.26
Batch size:  30

Current batch:  145
Estimated batches:  206

Max. simulation time (sec):  0
Elapsed simulation time:  25s

Number of transitions:  6
Min. number of firing for each transition:  0
Number of transitions that have fired at least once:  4
Number of transitions that have reached the min. number of firings:  4
Number of non-fired transitions:  2
Number of transitions firings:  4352

Confidence level %:  95.0
Max. relative error %:  1.0
Does it have reached the max. relative error?  No
```

```
Batch:  129      Error %:  8.804509063256237
Batch:  130      Error %:  8.770580193070293
Batch:  131      Error %:  8.73704056661038
Batch:  132      Error %:  8.703882797784892
Batch:  133      Error %:  8.6710996952412
Batch:  134      Error %:  8.638684255813601
Batch:  135      Error %:  8.606629658238704
Batch:  136      Error %:  8.574929257125442
Batch:  137      Error %:  8.54357657716761
Batch:  138      Error %:  8.512565307587487
Batch:  139      Error %:  8.481889296799709
Batch:  140      Error %:  8.451542547285165
Batch:  141      Error %:  8.42151921066519
Batch:  142      Error %:  8.391813582966892
Batch:  143      Error %:  8.362420100070908
```

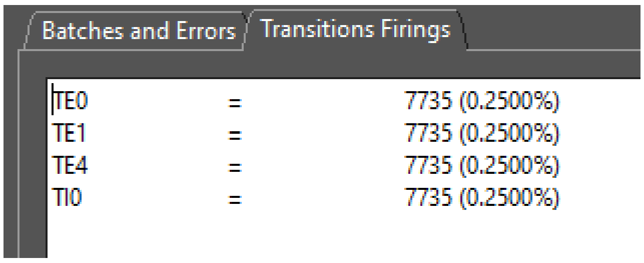
Figure 40: Stationary Simulator

The information displayed in this window is self-describing. The stationary simulator has two tabs. The *Batches and Errors* tab displays the history of the batches processed as well as the relative error of the simulation up to that point (see Figure 41). The *Transitions Firings* tab shows the number of firings for each fired transition as well as the percentage of firing in relation to the other fired transitions (see Figure 42).



```
Batch:  3715     Error %:  1.640667559372358
Batch:  3716     Error %:  1.6404467873119666
Batch:  3717     Error %:  1.6402261043504096
Batch:  3718     Error %:  1.6400055104277727
Batch:  3719     Error %:  1.6397850054841971
Batch:  3720     Error %:  1.6395645894598825
Batch:  3721     Error %:  1.6393442622950822
Batch:  3722     Error %:  1.639124023930107
Batch:  3723     Error %:  1.6389038743053237
Batch:  3724     Error %:  1.6386838133611554
Batch:  3725     Error %:  1.638463841038081
Batch:  3726     Error %:  1.638243957276635
Batch:  3727     Error %:  1.6380241620174076
Batch:  3728     Error %:  1.637804455201046
```

Figure 41: Batches and Errors



```
TE0          =          7735 (0.2500%)
TE1          =          7735 (0.2500%)
TE4          =          7735 (0.2500%)
TI0          =          7735 (0.2500%)
```

Figure 42: Transitions Firings

The stationary simulation ends when the maximum relative error is reached, or when the maximum simulation time is reached, or the minimum number of firings for each transition is reached, whatever occurs first. There is a progress bar on the top of that window that shows the progress of the simulation. This simulation progress undergoes adjustments depending on the simulation, as the estimated number of batches to reach the relative error can be re-estimated, thus changing the progress of the simulation. In addition, the user can pause and continue the simulation, as well as stop it at any time (see Figure 43).

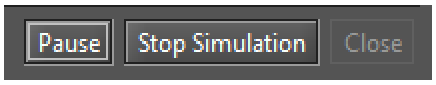


Figure 43: Stationary Simulator Buttons

When the simulation ends the user can export the simulation result as plain text or as a spreadsheet (MS Excel) (see Figure 44). A large number of statistics are computed considering the result of the simulation.

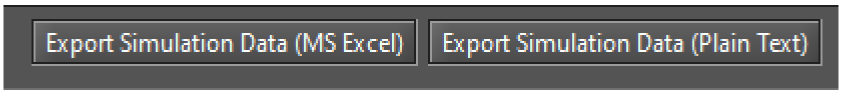

Export Simulation Data (MS Excel)
Export Simulation Data (Plain Text)

Figure 44: Export Buttons

Some statistics generated by the simulator:

- Sample Size
- Mean
- Midrange
- Minimum
- 1st Quartile
- 2nd Quartile
- 3rd Quartile
- Maximum
- IQR (interquartile range)
- Range
- RMS (root mean square)
- Variance
- Standard Deviation
- Mean Absolute Deviation
- Coeff. Of Variation
- Sum
- Sum of Squares
- Skewness
- Kurtosis
- Standard Error
- Relative Error

At the end of the simulation, the result is shown in the Result tab of the main window. Figure 45 demonstrates the example model we have used in the simulator. Listing 1 shows an example of the output generated by the simulator. In this example, only one metric has been evaluated.

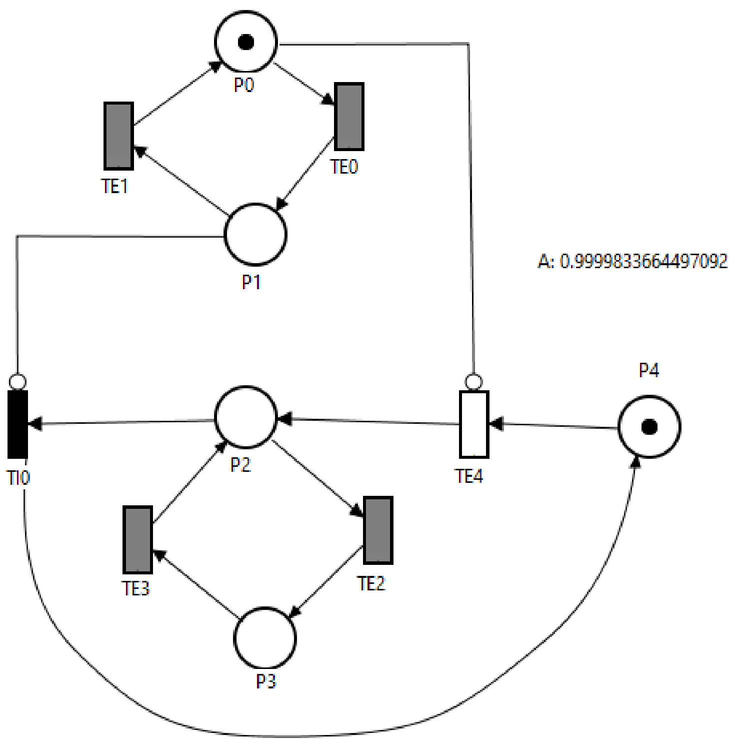


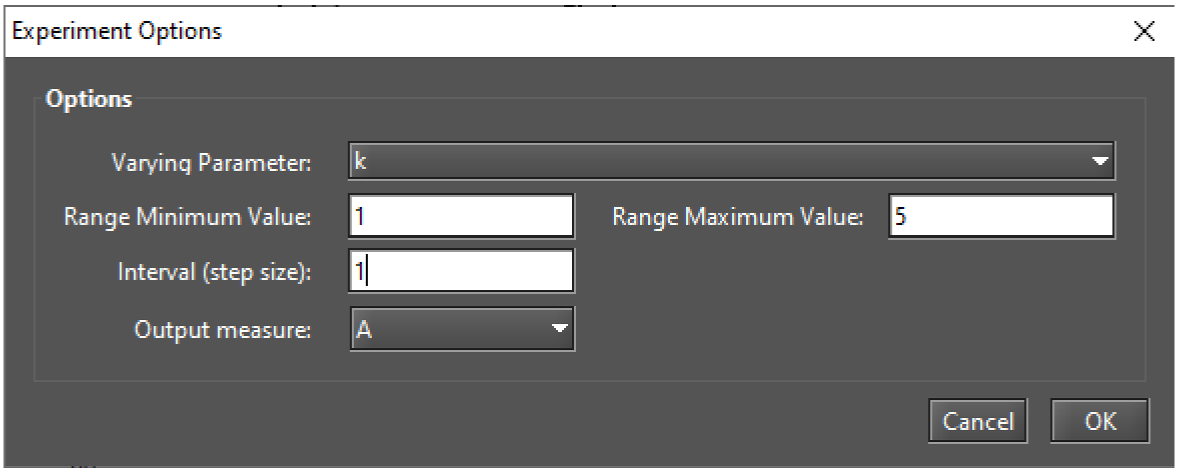
Figure 45: SPN Model for Simulation

Listing 1: Stationary Simulation Result

```
STATIONARY SIMULATION RESULT
————————————————————————————————

Confidence Level %: 95.0
Max. Relative Error %: 1.0
Min. Firing for each Transition: 0
Max. Simulation Time: 0
Min. Warm-up Period: 50


Warm-up Period: 5135.85
Global Time: 453749665.85


Batch Size: 30
Batches: 10000
Transitions Firings: 300001
Fired Transitions: 4
Non-Fired Transitions: 2


Fired Transitions
```

```
TE0      =        75001  (0.2500%)
TE1      =        75000  (0.2500%)
TE4      =        75000  (0.2500%)
TI0      =        75000  (0.2500%)

Non−Fired Transitions
TE2
TE3

Descriptive Statistics
______________________________________________
Metric : A, P{(#P0>0)OR(#P2>0)}
Result: 0.9999833664497092
Nines: 4.7790150443977675
Confidence Interval: [0.9999833628996808,0.9999833699997376]
Standard Error: 1.811053049610908E−9
Error %: 1.0
Sample Size, n: 10000
Midrange: 0.9999817534036552
Minimum: 0.9999784761307453
1st Quartile: 0.9999833204467997
2nd Quartile: 0.9999833471453827
3rd Quartile: 0.9999833889670042
Maximum: 0.9999850306765653
IQR: 6.852020451031393E−8
Range: 6.5545458199922635E−6
RMS: 0.9999833664497279
Variance, s^2: 3.2799131485049696E−14
Standard Deviation, s: 1.811053049610908E−7
Mean Absolute Deviation: 6.560655693113038E−8
Coeff. Of Variation: 1.8110831743539695E−7
Sum: 9999.833664497091
Sum Sq: 9999.667331761308
Skewness: −11.6910502101944
Kurtosis: 233.18498785918288
```

Now, we will demonstrate an example of running the stationary simulation with the “Experiment” parameter active. Figure 46 shows the dialog box that appears when confirming the input parameters for the simulation and checking the “Experiment” option.


Experiment Options

Options

Varying Parameter: k

Range Minimum Value: 1

Range Maximum Value: 5

Interval (step size): 1

Output measure: A

Figure 46: Stationary Simulation with Experiment

In this new window, the user must select the variable that will be changed, its minimum and maximum value as well as the step size. In addition, the user must select the metric that will be considered. At the end of the simulation, a graph will be plotted considering the metric value for each change in the variable. As we can see in Figure 47, the mean value of the selected metric is presented as well as its confidence intervals for each variation in the variable. When placing the mouse over the point that represents the mean, the values of the confidence intervals are displayed as a hint.

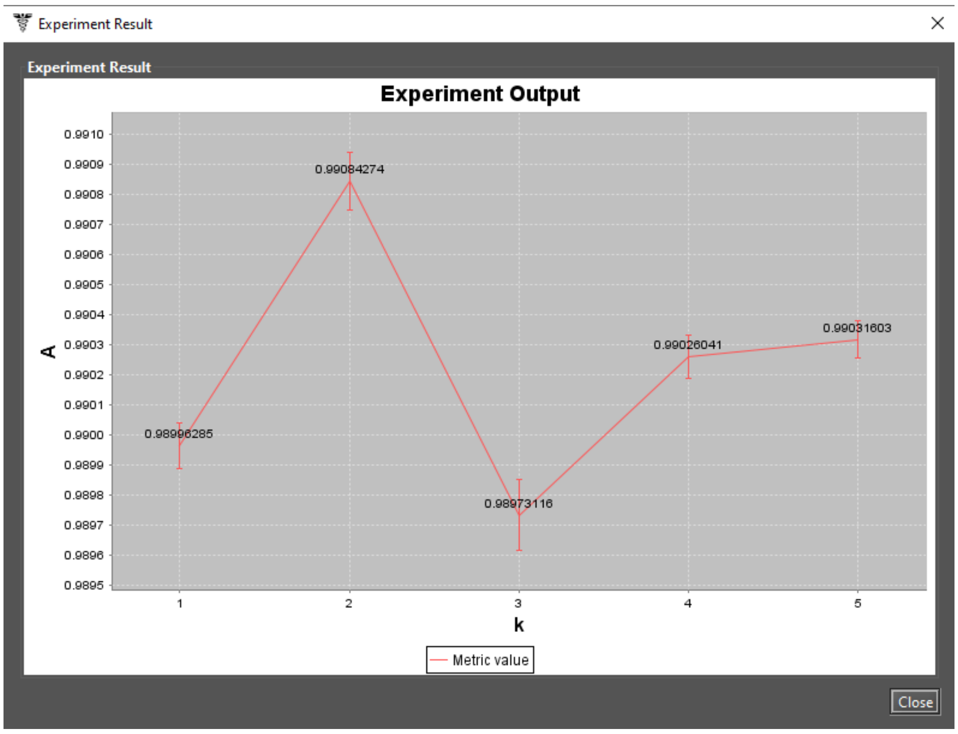
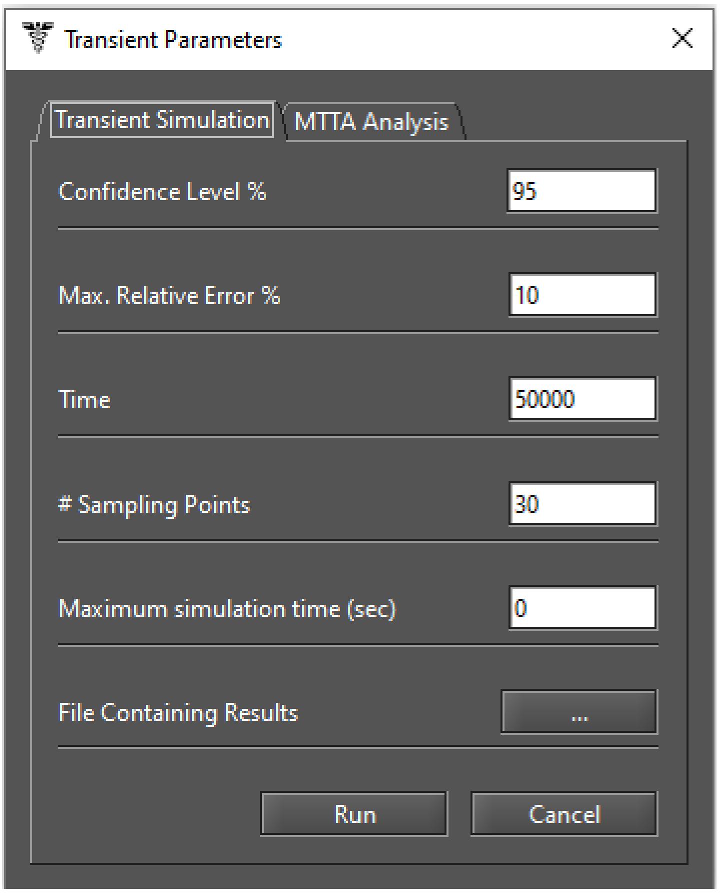


Figure 47: Stationary Simulation Experiment Output for an SPN Model

2.1.2 Transient Simulation

Figure 48 shows the "Transient Simulation" input window. The parameters are detailed as follows.

- **Confidence Level.** The confidence interval for obtaining the metrics.
- **Max. Relative Error.** Sets the maximum percentage of relative error.
- **Simulation Time.** Sets the simulation time.
- **Sampling Points.** Sets the number of sampling points that will be considered in order to obtain the results.
- **Max simulation real time.** It is used to define the maximum simulation time. This time corresponds to the physical time and must be defined in seconds.
- **File Containing Results.** Select the file to save simulation results.



Confidence Level % 95

Max. Relative Error % 10

Time 50000

Sampling Points 30

Maximum simulation time (sec) 0

File Containing Results

Figure 48: Transient Simulation

2.1.3 MTTA Simulation

Mercury provides a feature in the transient simulation that makes it possible to evaluate the behavior of absorbing models through simulation and from there generate a large number of results. Figure 49 shows an example of an absorbing model for simulation.

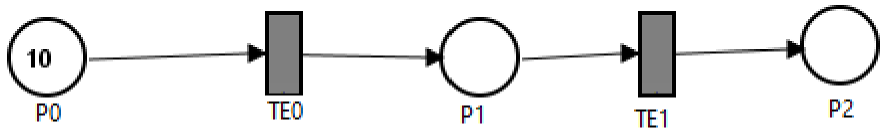


Figure 49: SPN Absorbing Model

The mean time to absorption (MTTA) simulation is accessed by following the menu depicted in Figure 50.

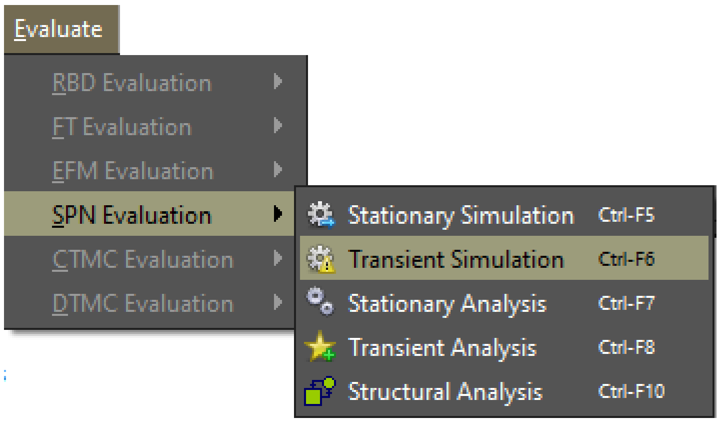
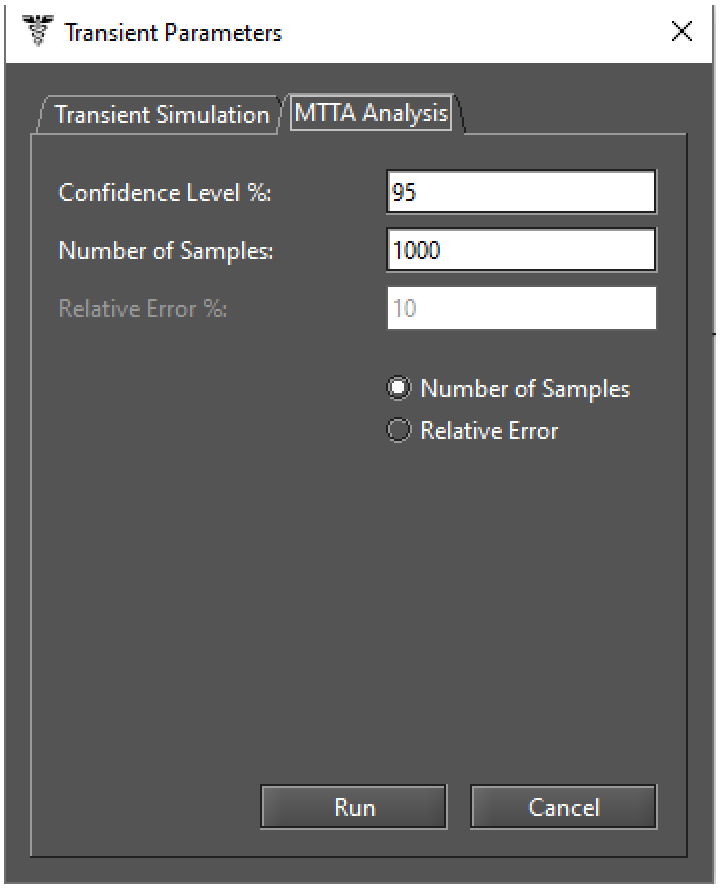

Evaluate
RBD Evaluation
FT Evaluation
EFM Evaluation
SPN Evaluation
CTMC Evaluation
DTMC Evaluation
Stationary Simulation Ctrl-F5
Transient Simulation Ctrl-F6
Stationary Analysis Ctrl-F7
Transient Analysis Ctrl-F8
Structural Analysis Ctrl-F10

Figure 50: Accessing the Transient Simulator

When accessing this menu, a window with two tabs appears (see Figure 51). The first tab contains the input parameters for the default transient simulator. The second tab ("MTTA Analysis") shows the input parameters for MTTA simulation, and these parameters are described below:

- **Confidence Level %.** The confidence interval for generating the statistics.
- **Number of Samples.** The total number of samples that the simulator will collect. After collecting the samples, the MTTA simulator generates statistics considering these data.
- **Relative Error %.** The maximum relative error to be considered. The MTTA simulation only stops when the relative error of the simulation is equal to or below the relative error defined by the user.


Transient Parameters

Transient Simulation | MTTA Analysis

Confidence Level %: 95

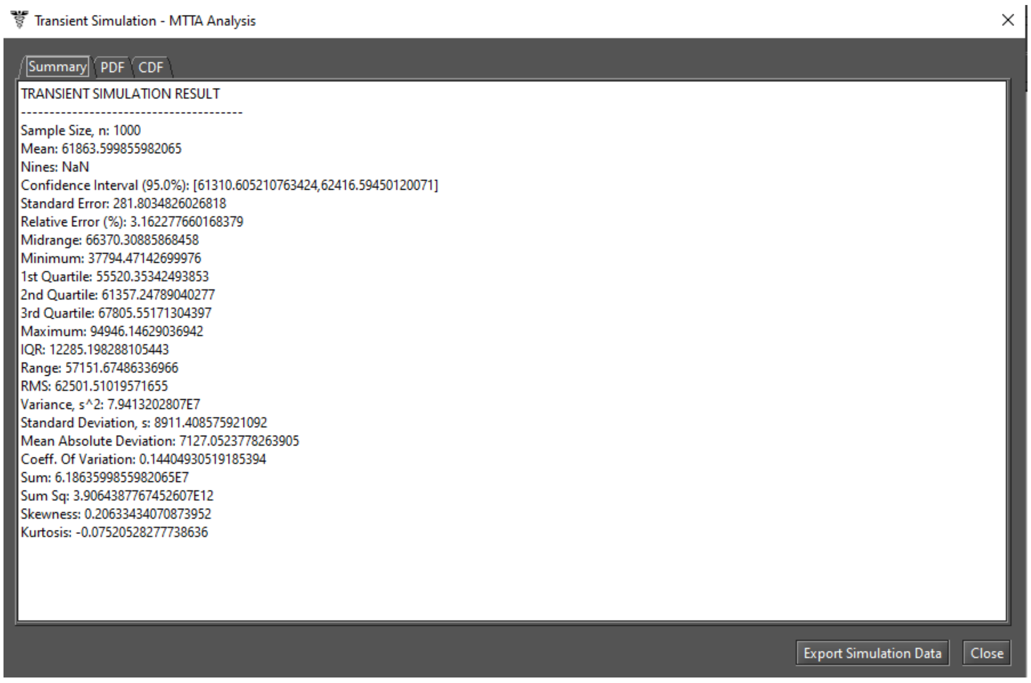
Number of Samples: 1000

Relative Error %: 10

(●) Number of Samples
( ) Relative Error

Figure 51: MTTA Analysis Dialog

At the end, a window shows the results of the transient simulation for the absorbing model under evaluation. In the Summary tab are presented the statistics about the simulation. As we can see in the Figure 52, a large number of statistics are presented. Listing 2 shows the output of an MTTA simulation in detail.


Transient Simulation - MTTA Analysis

```
TRANSIENT SIMULATION RESULT
---------------------------------------
Sample Size, n: 1000
Mean: 61863.599855982065
Nines: NaN
Confidence Interval (95.0%): [61310.605210763424,62416.59450120071]
Standard Error: 281.8034826026818
Relative Error (%): 3.162277660168379
Midrange: 66370.30885868458
Minimum: 37794.47142699976
1st Quartile: 55520.35342493853
2nd Quartile: 61357.24789040277
3rd Quartile: 67805.55171304397
Maximum: 94946.14629036942
IQR: 12285.198288105443
Range: 57151.67486336966
RMS: 62501.51019571655
Variance, s^2: 7.9413202807E7
Standard Deviation, s: 8911.408575921092
Mean Absolute Deviation: 7127.0523778263905
Coeff. Of Variation: 0.14404930519185394
Sum: 6.1863599855982065E7
Sum Sq: 3.9064387767452607E12
Skewness: 0.20633434070873952
Kurtosis: -0.07520528277738636
```

Figure 52: MTTA Result - Summary

Listing 2: MTTA Result

```
TRANSIENT SIMULATION RESULT
______________________________________________

Sample Size, n: 1000
Mean: 61863.599855982065
Nines: NaN
Confidence Interval (95.0%): [61310.605210763424,62416.59450120071]
Standard Error: 281.8034826026818
Relative Error (%): 3.162277660168379
Midrange: 66370.30885868458
Minimum: 37794.47142699976
1st Quartile: 55520.35342493853
2nd Quartile: 61357.24789040277
3rd Quartile: 67805.55171304397
Maximum: 94946.14629036942
IQR: 12285.198288105443
Range: 57151.67486336966
RMS: 62501.51019571655
Variance, s^2: 7.9413202807E7
Standard Deviation, s: 8911.408575921092
Mean Absolute Deviation: 7127.0523778263905
Coeff. Of Variation: 0.14404930519185394
Sum: 6.1863599855982065E7
Sum Sq: 3.9064387767452607E12
Skewness: 0.20633434070873952
Kurtosis: −0.07520528277738636
```

The PDF tab displays the probability distribution function of the generated data (see Figure 53). The cumulative distribution function of these data is shown in the CDF tab (see Figure 54). When placing the cursor on any blue point on the plotted curve, the tool shows the values of the x and y axes as a hint (see Figure 55). The user has also an option to export the results to an MS Excel spreadsheet (a .xls file).

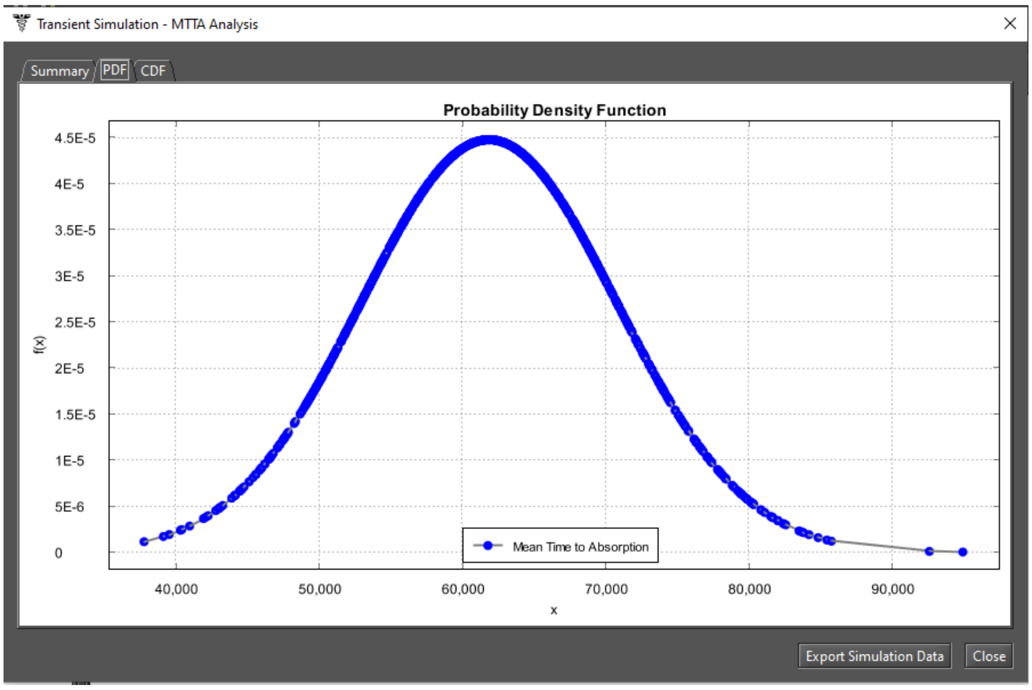


Figure 53: MTTA Result - PDF

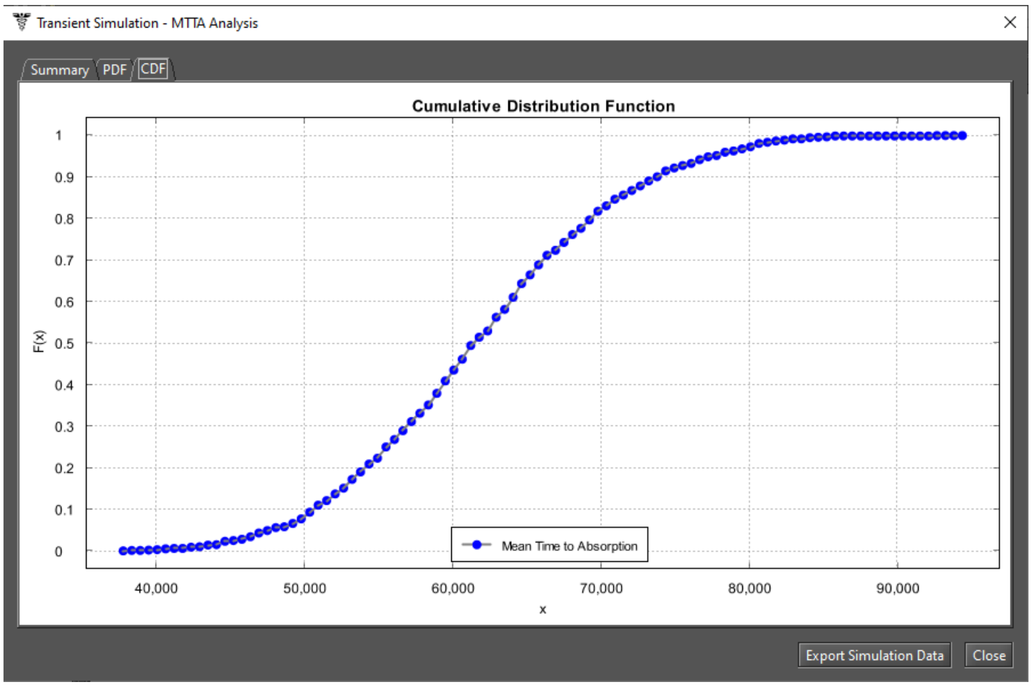


Figure 54: MTTA Result - CDF

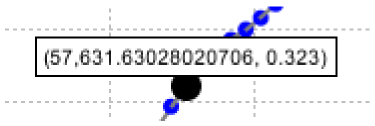


Figure 55: X and Y Axis Values

2.2 SPN Analysis

The **Stationary Analysis** and **Transient Analysis** both compute results by generating the underlying CTMC related to the state space of the SPN under evaluation. Stationary analysis computes steady-state probabilities, useful for evaluating the long-term average behavior of modeled systems. On the other hand, transient analysis computes time-dependent probabilities, useful for evaluating the behavior of modeled systems at a particular point in time.

2.2.1 Stationary Analysis

Figure 56 shows the "Stationary Analysis" window, which has a combo box for selecting one of two solution methods available: **Direct - GTH** (Grassmann-Taksar-Heyman) or **Iterative - Gauss-Seidel**.

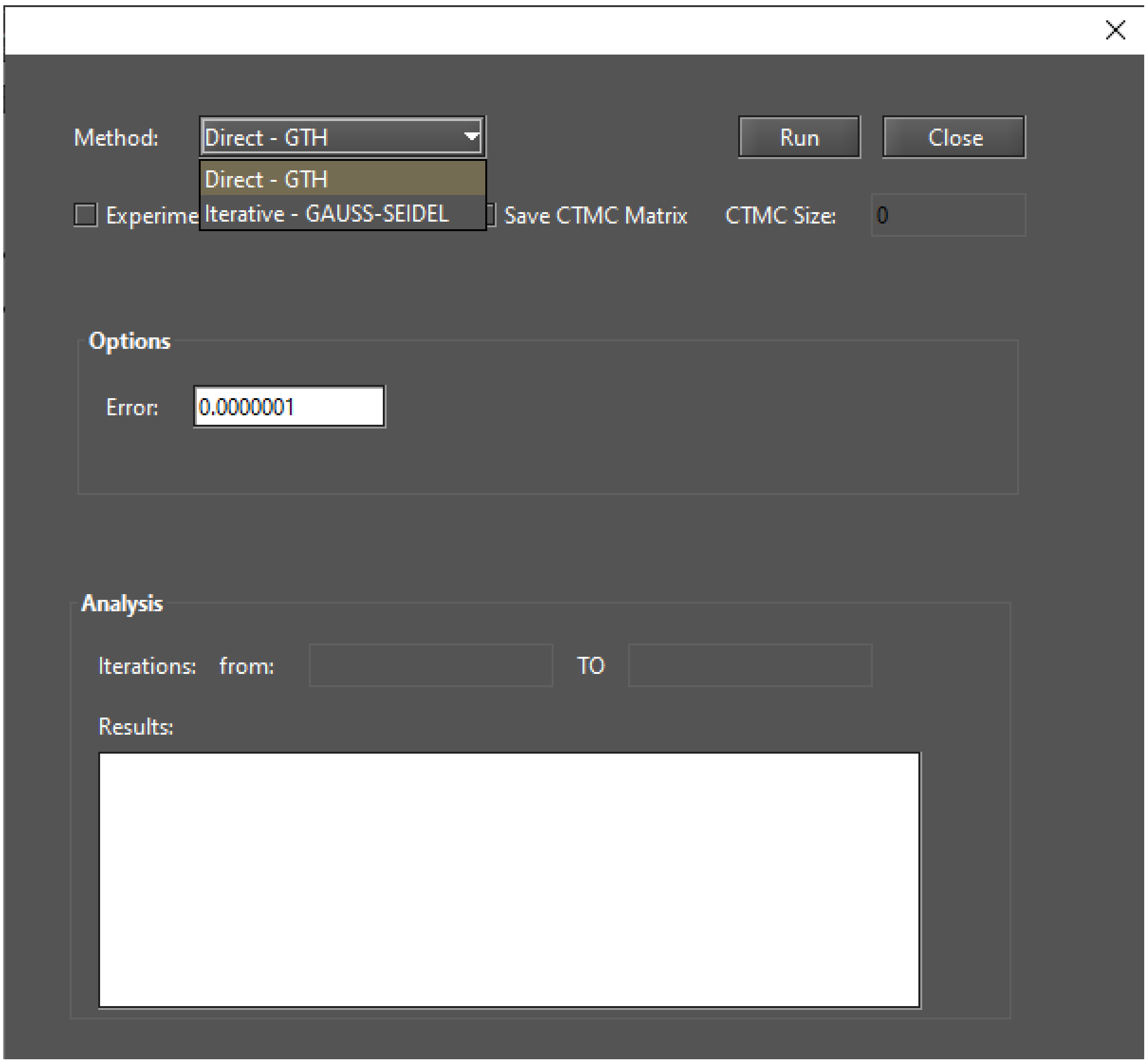


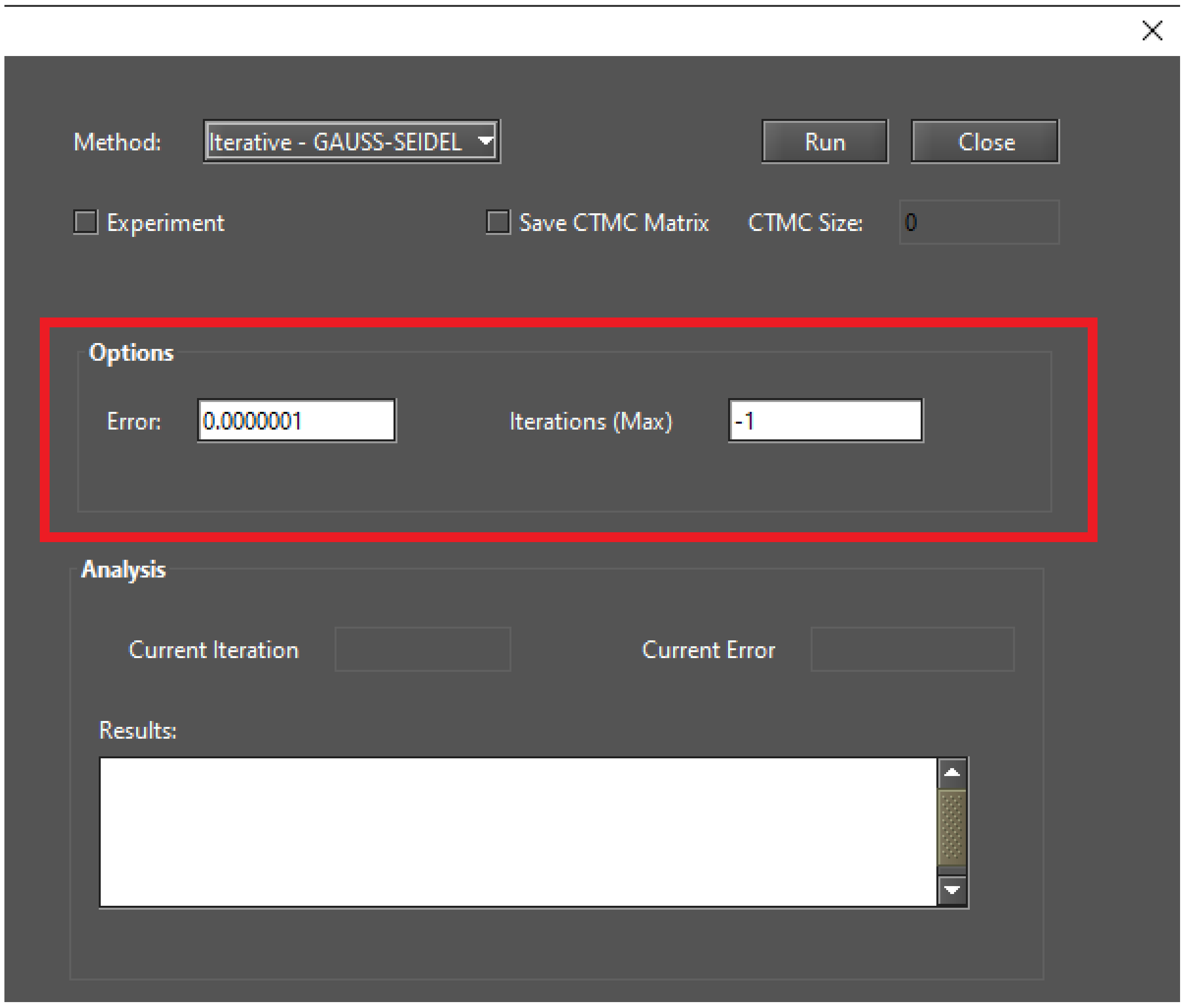
Figure 56: Stationary Analysis Window

When solving the model through GTH, the user can change the **maximum error** used in the algorithm. The default value for the maximum error is 0.0000001 ($10^{-}7$). By clicking on the "Run" button, the solution algorithm is triggered and as soon as it finishes, the results are presented in the text area at the bottom of the window (see Listing 3).

Listing 3: Stationary Analysis for an SPN

```
Performing stationary analysis...
Generating CTMC... Tue Feb 11 07:01:25 BRT 2020
CTMC generated...
Executing numerical method... Tue Feb 11 07:01:25 BRT 2020
Done! Tue Feb 11 07:01:25 BRT 2020
S0=0.9903691816162996
S1=0.009630818383700439
```

When solving the model through Gauss-Seidel, besides the maximum error, the user can also change the maximum number of iterations. The default value for such a parameter is "-1", indicating that the algorithm only stops when the convergence of results is reached, considering the entered error (see Figure 57).



Method: Iterative - GAUSS-SEIDEL

CTMC Size: 0

Error: 0.0000001

Iterations (Max) -1

Figure 57: Stationary Analysis Window - Gauss-Seidel Method

The metrics are updated as soon as the analysis is completed, independently of the chosen method, their values are updated in the SPN drawing area, as depicted by Figure 58, where a metric named "Availability" is presented.

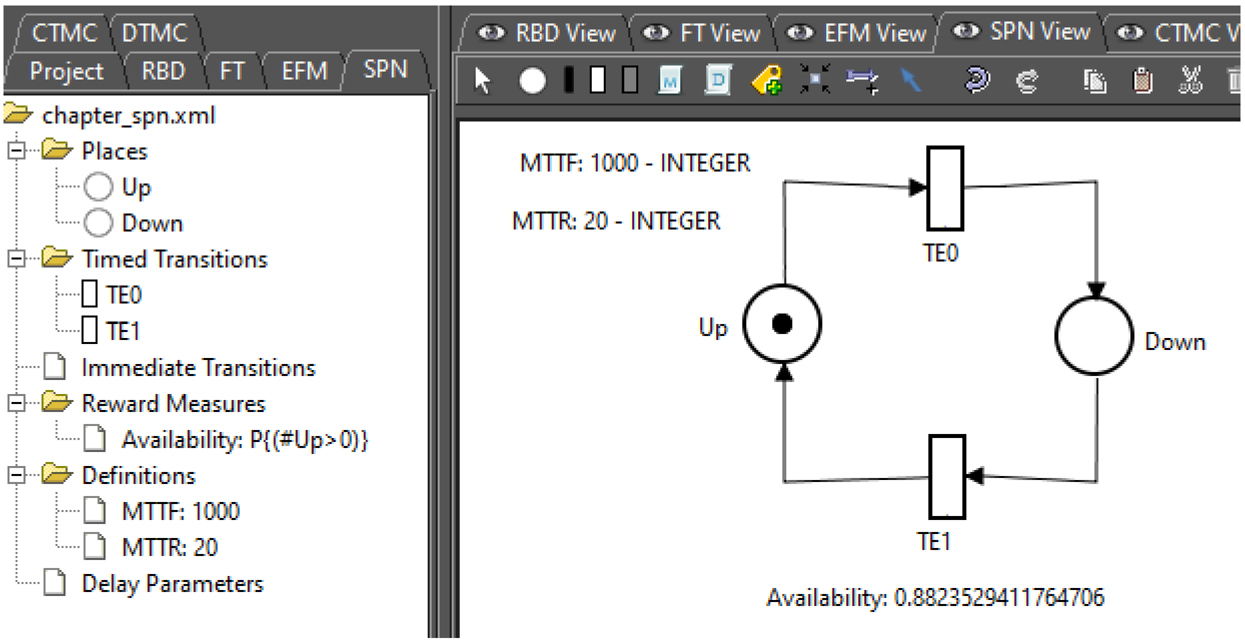
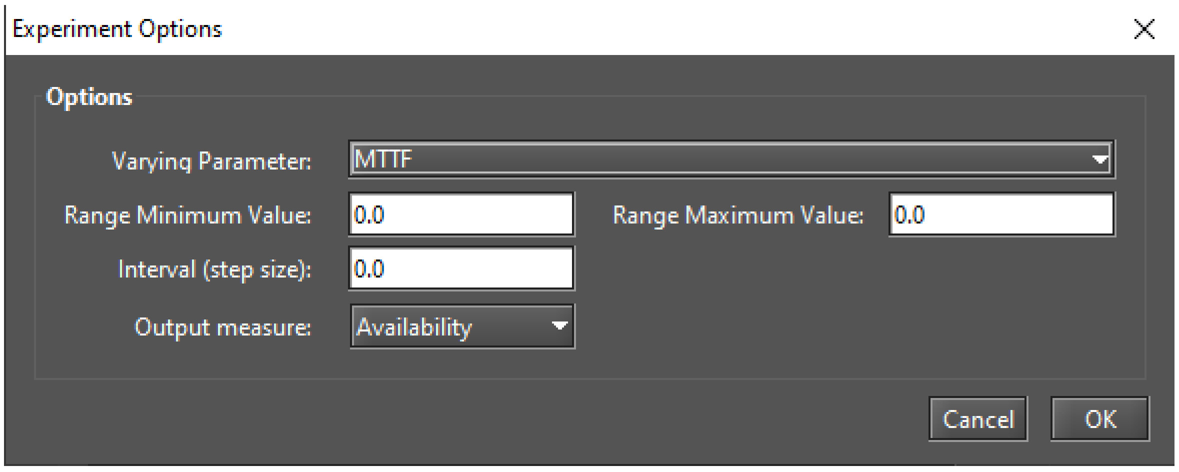


Figure 58: An SPN Model

The SPN model can also be solved for a range of values of user-defined parameters. This is accomplished by clicking in the "Experiment" checkbox on the "Stationary Analysis" window and then on the "Run" button. A new dialog is displayed in order to define the options for the experiment to be carried out (see Figure 59).



Experiment Options

Options

Varying Parameter: MTTF

Range Minimum Value: 0.0

Range Maximum Value: 0.0

Interval (step size): 0.0

Output measure: Availability

Figure 59: SPN Experiment

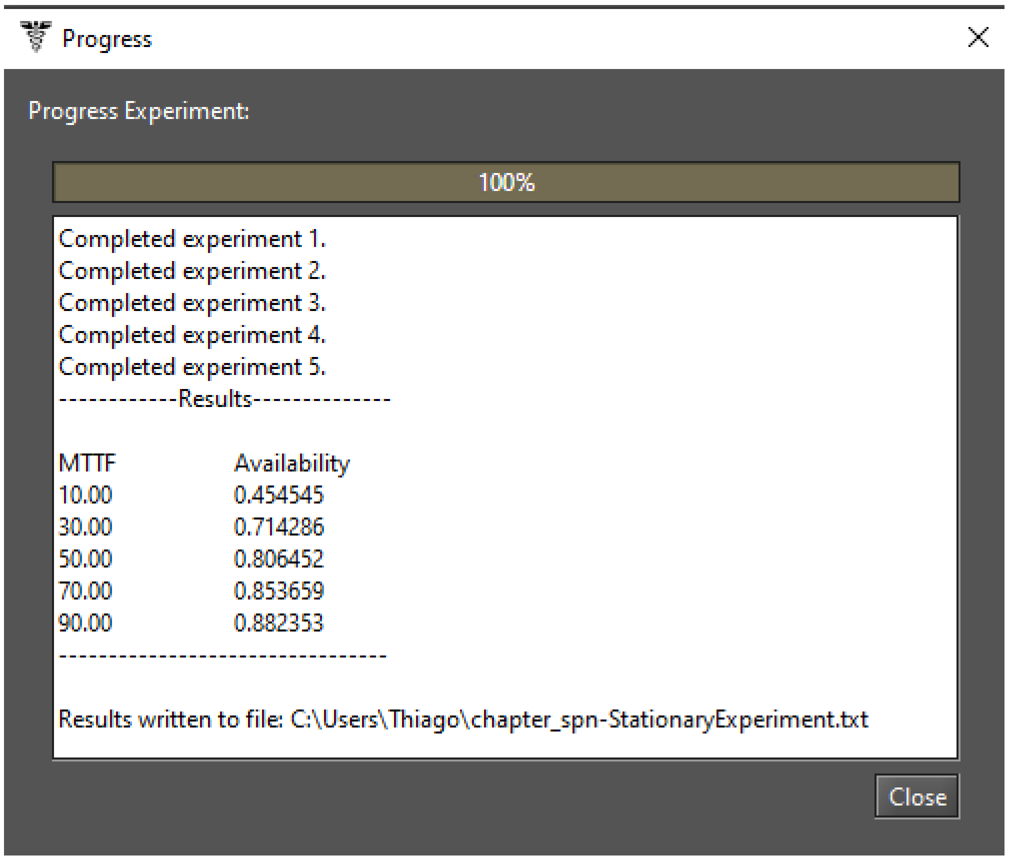
Below, we describe each of them.

- **Varying Parameter.** The parameter to have its value changed.

- **Output measure.** The measure to be evaluated.

- **Range Minimum Value.** The initial value to be assigned to the selected parameter.
- **Range Maximum Value.** The final value to be assigned to the selected parameter.
- **Interval.** It is the step-size for changing the value of the parameter. The parameter starts with the minimum value and its value is increased by considering the entered interval. At each change, the selected measure is evaluated. The experiment ends when the maximum value for the parameter is reached.

At the end of the experiment, the results are shown and a graph is plotted, as we can see by looking at Figures 60 and 61, respectively.



```
Completed experiment 1.
Completed experiment 2.
Completed experiment 3.
Completed experiment 4.
Completed experiment 5.
------------Results--------------

MTTF          Availability
10.00         0.454545
30.00         0.714286
50.00         0.806452
70.00         0.853659
90.00         0.882353
---------------------------------

Results written to file: C:\Users\Thiago\chapter_spn-StationaryExperiment.txt
```

Figure 60: SPN Experiment Results

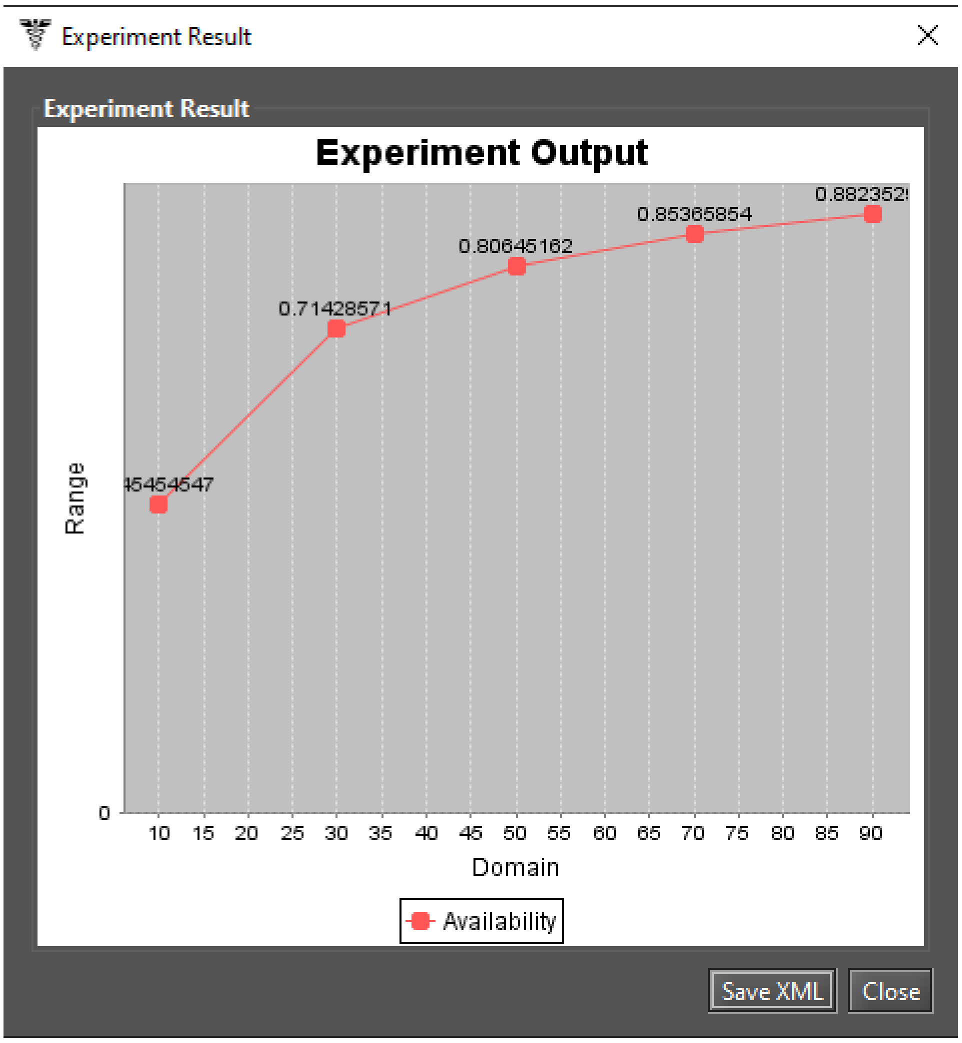
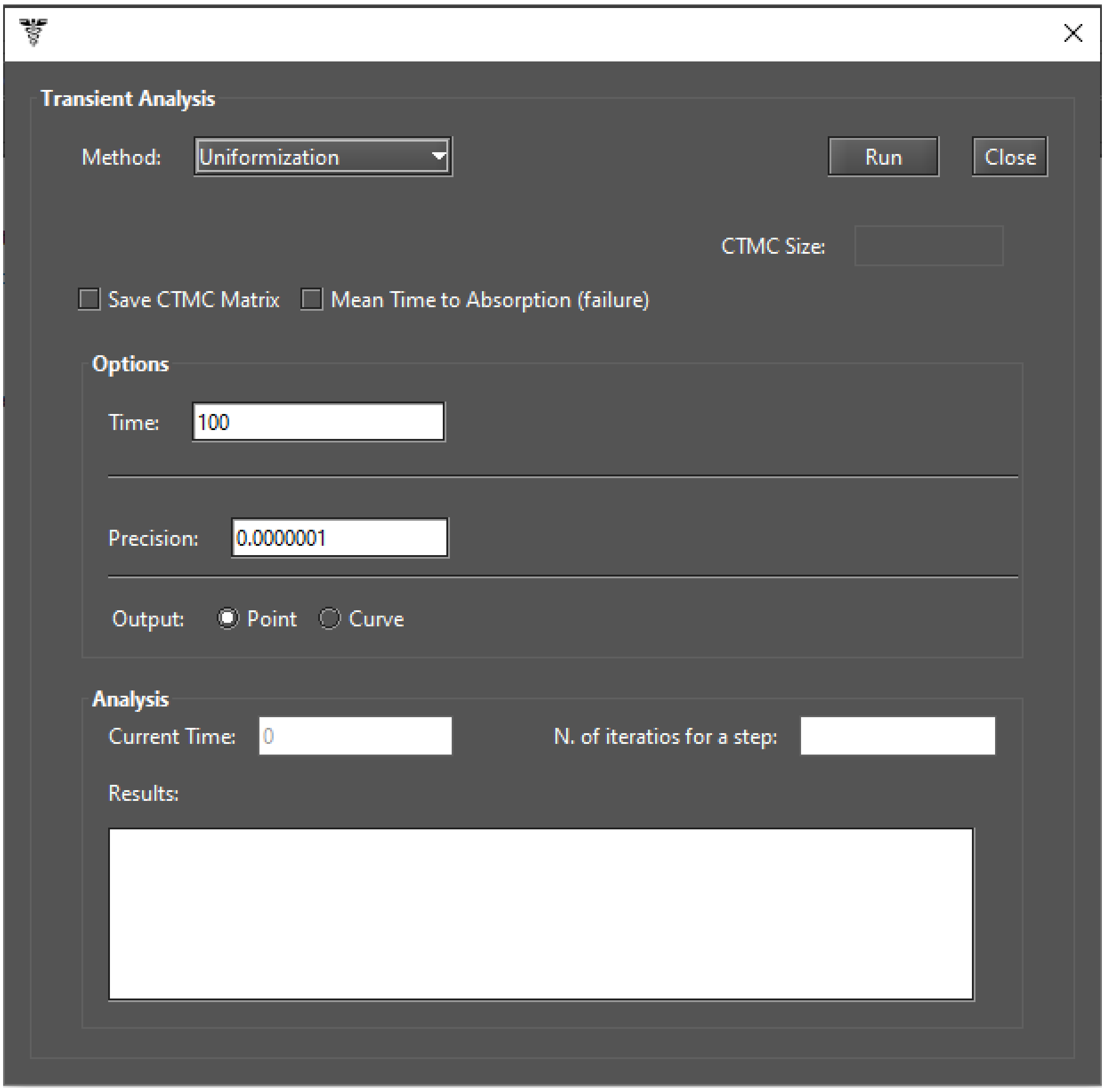


Figure 61: SPN Experiment Graph

2.2.2 Transient Analysis

Figure 62 shows the “Transient Analysis” window, which has a combo box for selecting one of the two solution methods available: **Uniformization** (also known as Jensen’s method) and **Runge-Kutta (4th order)**.


Transient Analysis

Method: Uniformization

CTMC Size:

Save CTMC Matrix [ ] Mean Time to Absorption (failure) [ ]

Options

Time: 100

Precision: 0.0000001

Output: [✓ Point] Curve

Analysis

Current Time: 0

N. of iteratios for a step:

Results:

Figure 62: Transient Analysis Window

When solving the model, the user can define:

- The **time** for which the analysis will be carried out (default: 100).
- the **precision** of results (default:10$^-$7),

When selecting the Uniformization method, note that the time required for obtaining the results is proportional to the time entered for the analysis because Uniformization is an iterative algorithm.

By clicking on the “Run” button, the solution algorithm is triggered. As soon as it finishes, the results are presented in the text area at the bottom of the “Transient Analysis” window, also they are written in a plain text file having the filename of the project appended with the “-TransientAnalysis.txt” suffix.

The “Transient Analysis” window also allows the user to choose between a **Point** or **Curve** analysis. **Point** analysis is the default, and it shows results only for the specific point in time. **Curve** analysis writes in a plain text file all measures valures computed in intermediate steps from time equals zero until the specified time.

The mean time to absorption (**MTTA**) is a metric that can be computed for absorbing SPNs in the “Transient Analysis” window, by checking the “Mean Time to Absorption (failure)” checkbox. The MTTA value is presented after the state probabilities in the “Results” text area.

2.3 SPN Structural Analysis

The Mercury tool provides a function for analyzing SPNs without generating the reachability graph. “Structural Analysis” makes it possible to prove some properties of a model by means of **invariants** and **traps** techniques. It is accessible in the menu Evaluate -> SPN Evaluation -> Structural Analysis (see Figure 63).

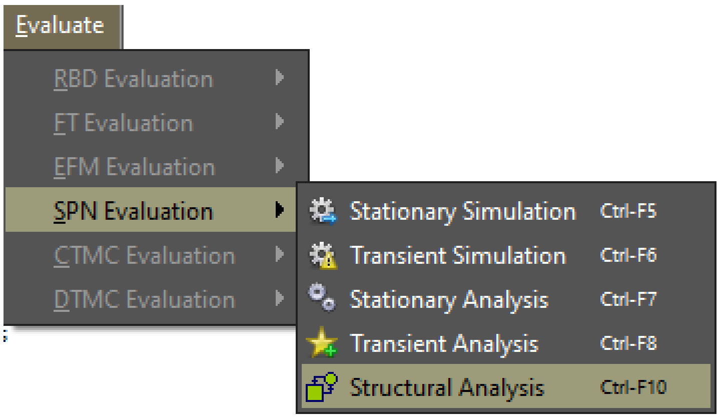
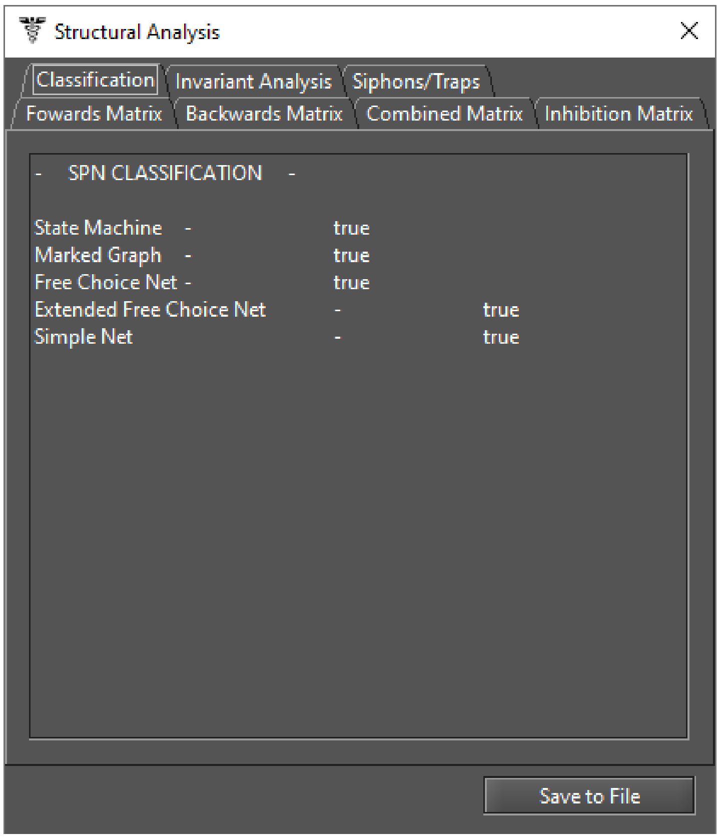

Evaluate
RBD Evaluation
FT Evaluation
EFM Evaluation
SPN Evaluation
CTMC Evaluation
DTMC Evaluation
Stationary Simulation Ctrl-F5
Transient Simulation Ctrl-F6
Stationary Analysis Ctrl-F7
Transient Analysis Ctrl-F8
Structural Analysis Ctrl-F10

Figure 63: Structural Analysis Function

When the structural analysis is finished, the “Structural Analysis” window is displayed containing various panels related to the SPN structural properties: **Forwards Matrix**, **Backwards Matrix**, **Combined Matrix**, **Inhibition Matrix**, **Classification**, **Invariant Analysis**, and **Siphons/Traps**. In this window, it is possible to save the results in a plain text file.



```
-   SPN CLASSIFICATION   -

State Machine    -          true
Marked Graph     -          true
Free Choice Net -           true
Extended Free Choice Net    -          true
Simple Net                  -          true
```

Figure 64: Structural Analysis Window

2.4 Token Game

Token Game makes it possible for the users to simulate the behavior of SPN models. In other words, users are able, for instance, to debug the model that is under analysis. Thus, mistakes in the construction of that model can be easily discovered and some improvements may be made in order to fix them. Figure 65 shows an SPN model with the Token Game functionality off.

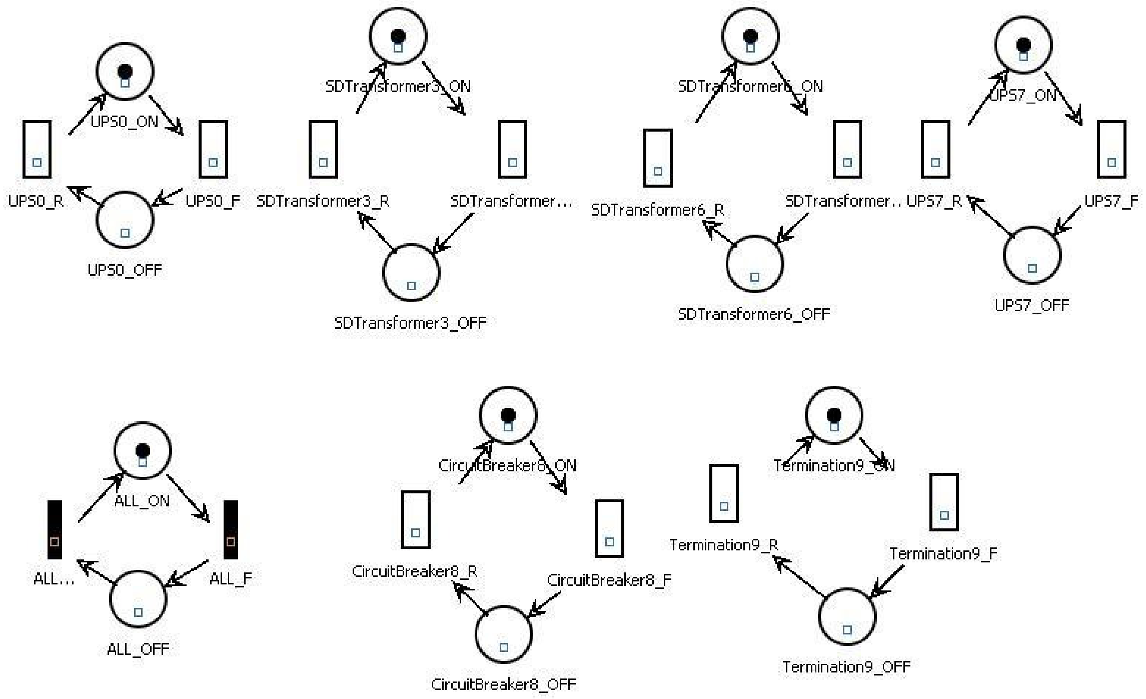


Figure 65: SPN Model

Considering the guard expression defined below, there is a transition *ALL_F* that represents the behavior of a system. In other words, when this transition is fired, the mode of that system is changed from operational to failure. The guard expression assigned to this transition (*ALL_F*) is represented as follows.

$$((\#Termination9\_ON = 0) OR (\#CircuitBreaker8\_ON = 0)$$

$$\text{OR } (((\#UPS0\_ON = 0) OR (\#SDTransformer3\_ON = 0))$$

$$\text{AND } ((\#UPS7\_ON = 0) OR (\#SDTransformer6\_ON = 0))))$$

By considering our SPN model and using the Token Game function, users can simulate equipment failures as well as the corresponding consequences on the system availability. In order to enable or disable this function, the user must click on the “Token Game” button highlighted in Figure 66. Moreover, when observing that figure, we can see that any component can fail through the fire of any enabled transition (the green’s ones on the figure).

Considering that the user has fired the transition “Termination9_F” (see Figure 67), this means that the termination has failed. So, we can see that there is only one transition ready to be fired (*ALL_F*), meaning that the system has changed to the failure mode, as expected.

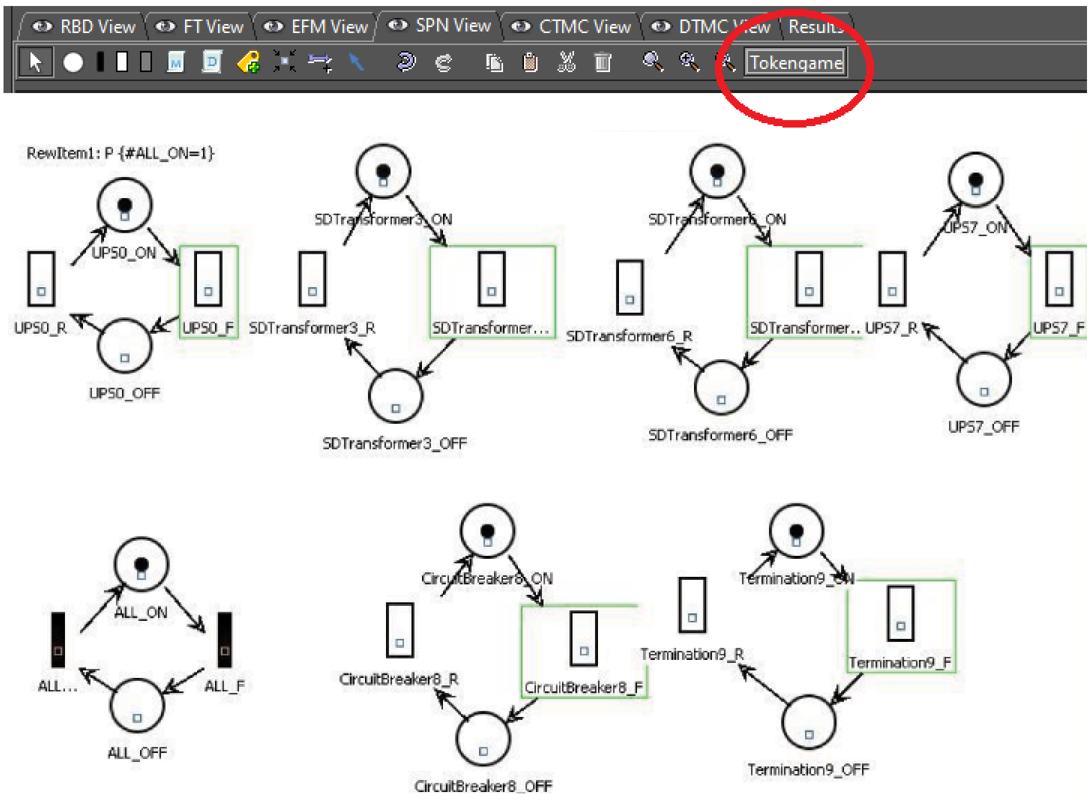


Figure 66: Activating the Token Game

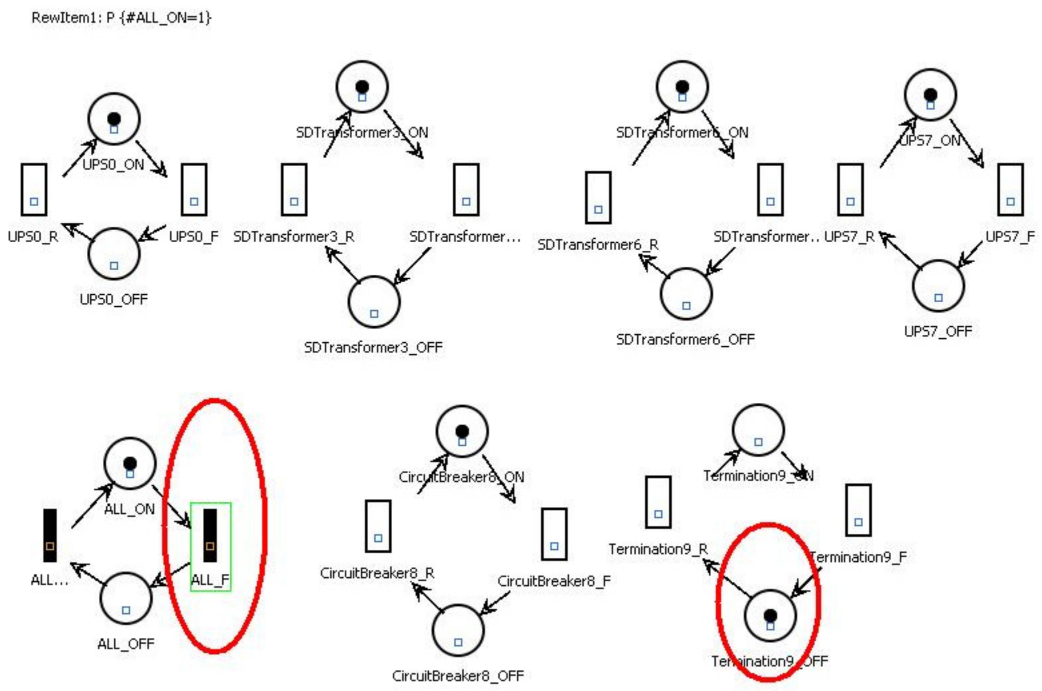
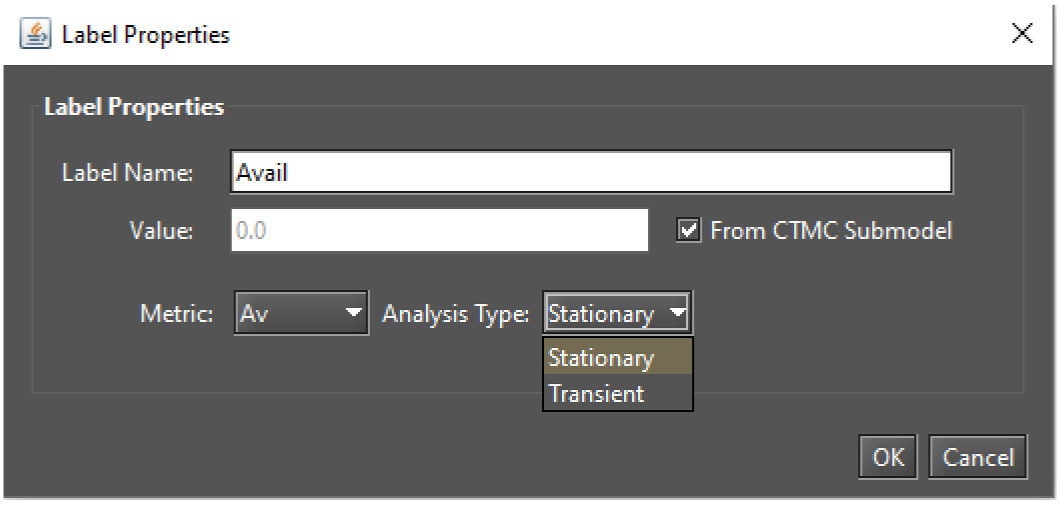


Figure 67: Token Game Example

2.5 Hierarchical Evaluation

The user can define the delays for SPN transitions by means of metrics computed from a CTMC model, which we call hierarchical composition. This is done through the use of SPN labels attached to the desired CTMC metric, as shown by Figure 68.



Label Properties

Label Name: Avail

Value: 0.0

[✓] From CTMC Submodel

Metric: Av

Analysis Type: Stationary

Stationary
Transient

Figure 68: Assigning a Metric from a CTMC Submodel to an SPN Label

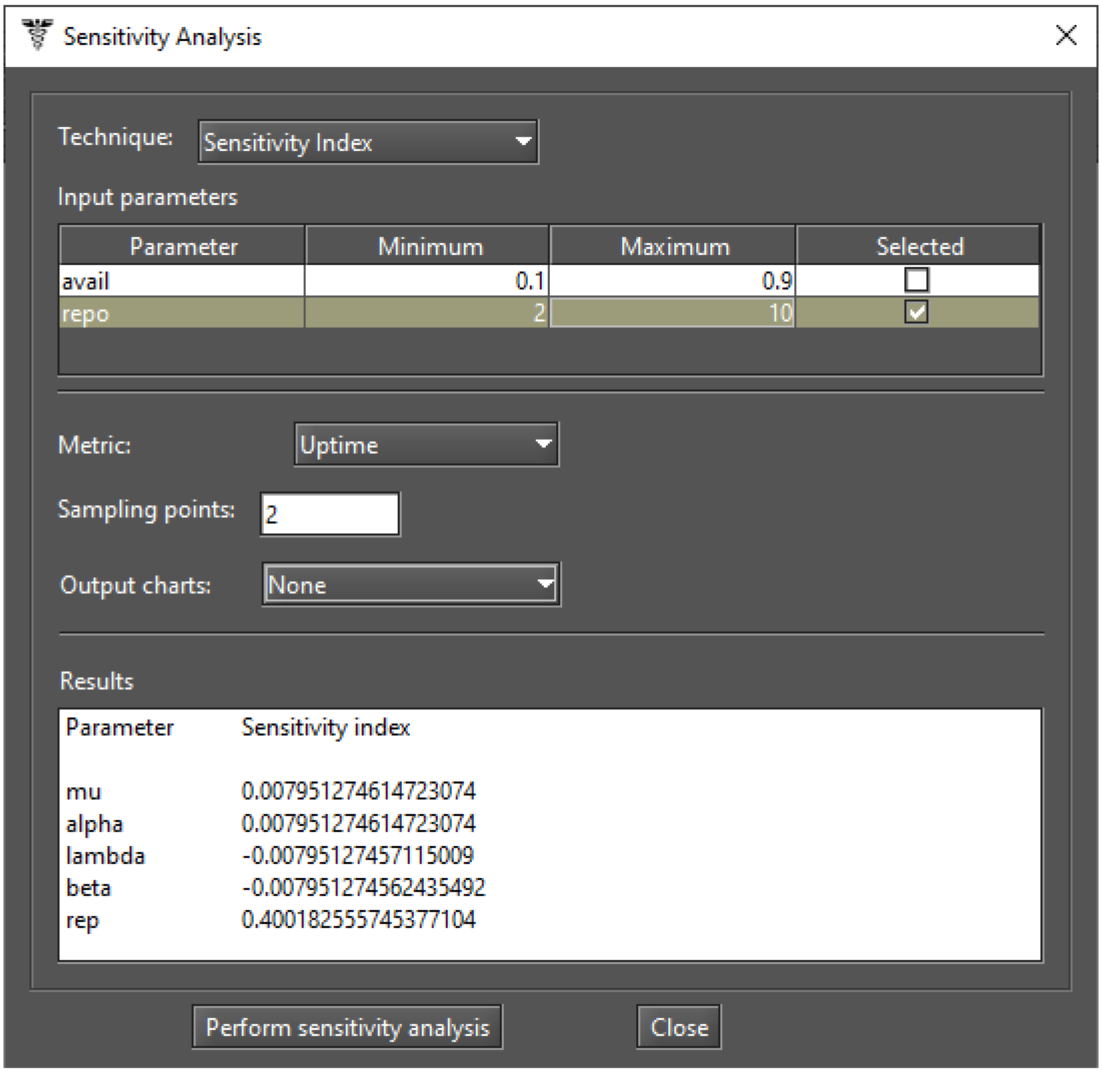
The user must check the box “From CTMC Submodel", and select one of the metrics previously defined in the CTMC. The type of analysis (Stationary or Transient) must be selected too. When the transient analysis is chosen, it is necessary to provide the time parameter.

2.6 Sensitivity Analysis

Mercury allows performing sensitivity analysis on SPNs that enables computing partial derivative sensitivity indices for them. This analysis is accessible in the menu Tools -> Sensitivity Analysis. Figure 69 shows the "Sensitivity Analysis" window.

In this window, the user must select between two sensitivity analysis methods: "Design of Experiments" or "Sensitivity Indices". The former uses the standard method of analysis of variance to identify the effect of each factor on the model results. The latter uses the technique of percentage difference, so it requires a minimum and maximum value for each parameter in order to compute the corresponding percentage variation on the chosen metric.

Mercury is able to compute sensitivity for an SPN measure with respect to each SPN delay parameter as well as with respect to the parameters of a CTMC sub-models that provide the value for any SPN transition.



Technique: Sensitivity Index

Input parameters

Parameter	Minimum	Maximum	Selected
avail	0.1	0.9	☐
repo	2	10	✓

Metric: Uptime

Sampling points: 2

Output charts: None

Results

```
Parameter      Sensitivity index

mu             0.007951274614723074
alpha          0.007951274614723074
lambda         -0.00795127457115009
beta           -0.007951274562435492
rep            0.400182555745377104
```

Figure 69: Sensitivity Analysis for an SPN Model

The results of the sensitivity analysis are presented in three possible output formats: "None", "JFreeChart", and "R". The user must select one of those outputs, as depicted by Figure 70. The default option is "None", which prints the results in the text area on the bottom of the “Sensitivity Analysis” window.

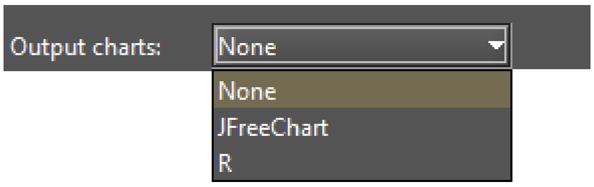


Figure 70: Output Options for Sensitivity Analysis

3 RBD Modeling and Evaluation

Reliability Block Diagram (RBD) is a success-oriented modeling approach and it makes it possible to create a visual representation of a system that shows how the components' reliability contributes to the failure or success of the system. When creating a new project, the default RBD model is created with an empty block. In the RBD view, the default model presents the empty block named b1. This block is gray and represents that its properties have not yet been defined, as shown by Figure 71. As we can see, RBD is evaluated from left to right.

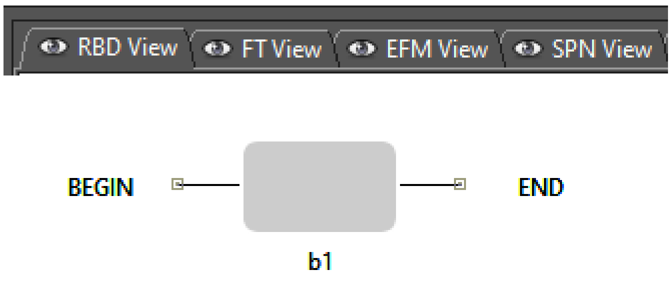


Figure 71: Initial RBD Model

In order to create a block, the user must right-click on the initial block. As we can see in Figure 72, on the pop-up menu that appears some options are available. Among the available options, the user can insert, edit, and remove blocks.

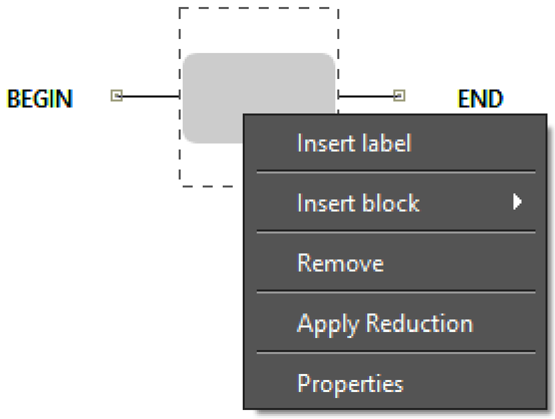


Figure 72: Block Popup Menu

To insert a block the user must go to the sub-menu item "Insert block" and, depending on how the block will be connected to the others, the option "Series" or "Parallel" must be selected. For each of these options, there are two types of blocks to be created: "Simple Block" and "k-out-of-n Block" (see Figure 73).

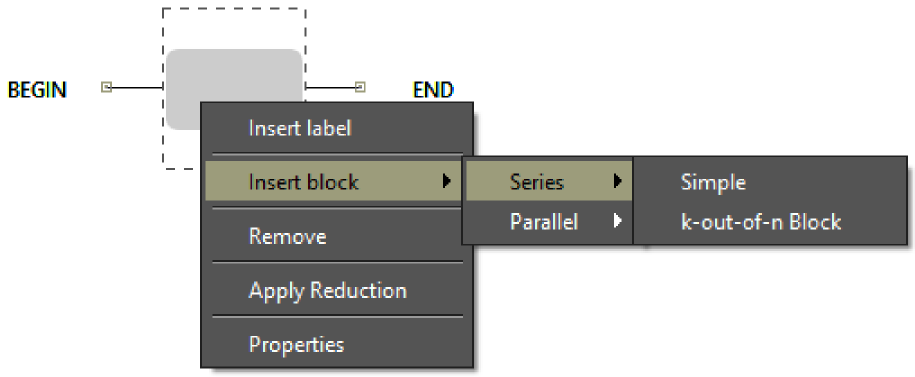
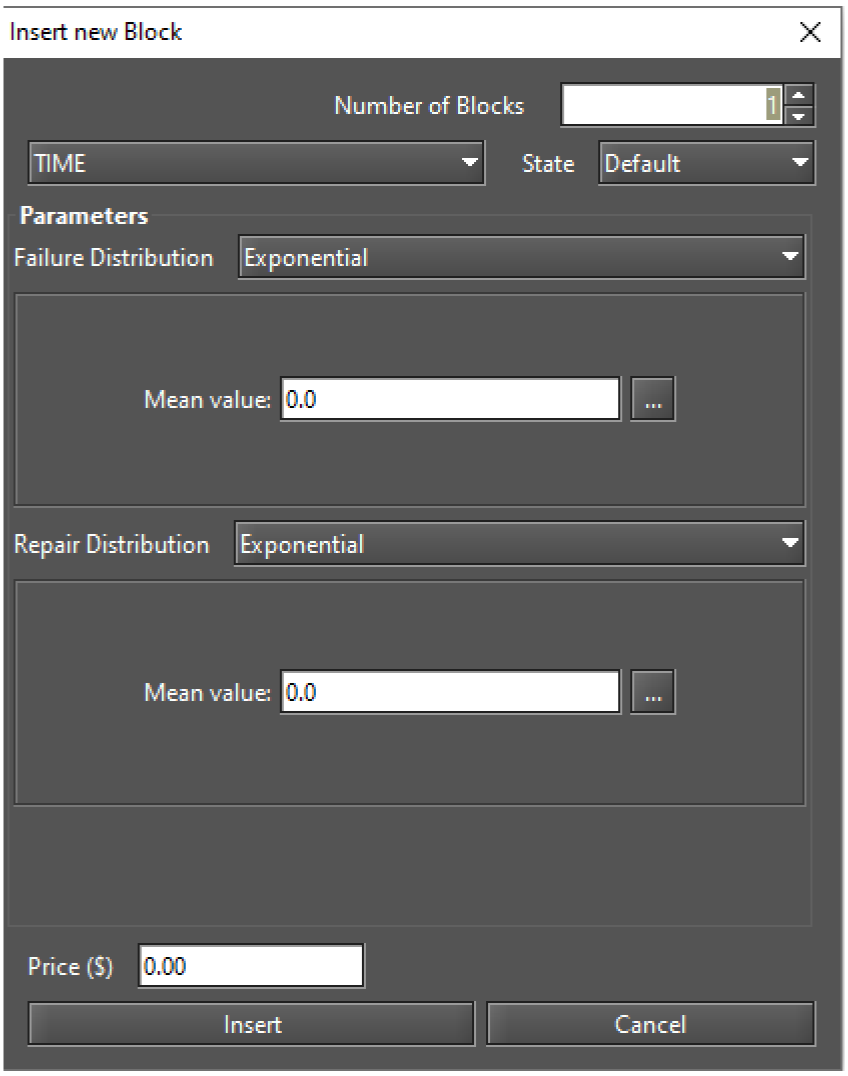


Figure 73: Inserting an RBD Block

After choosing the type of block to be inserted, a dialog box is displayed where the user can enter the block's properties. Figure 74 shows the dialog that appears when inserting a simple block, and Figure 75 shows the dialog that appears when inserting a k-out-of-n block.



Insert new Block

Number of Blocks: 1

TIME

State: Default

Parameters

Failure Distribution: Exponential

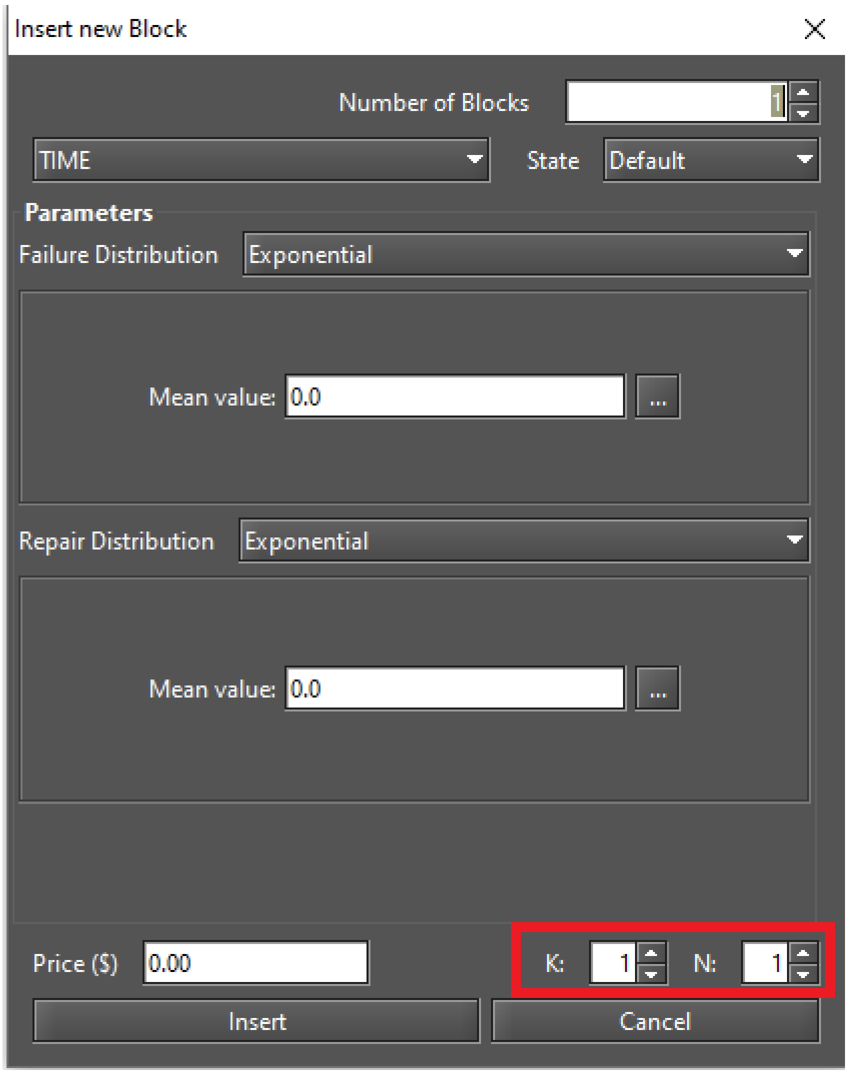
Mean value: 0.0

Repair Distribution: Exponential

Mean value: 0.0

Price ($): 0.00

Figure 74: Inserting a Simple Block


Insert new Block

Number of Blocks: 1

TIME

State: Default

Parameters

Failure Distribution: Exponential

Mean value: 0.0

Repair Distribution: Exponential

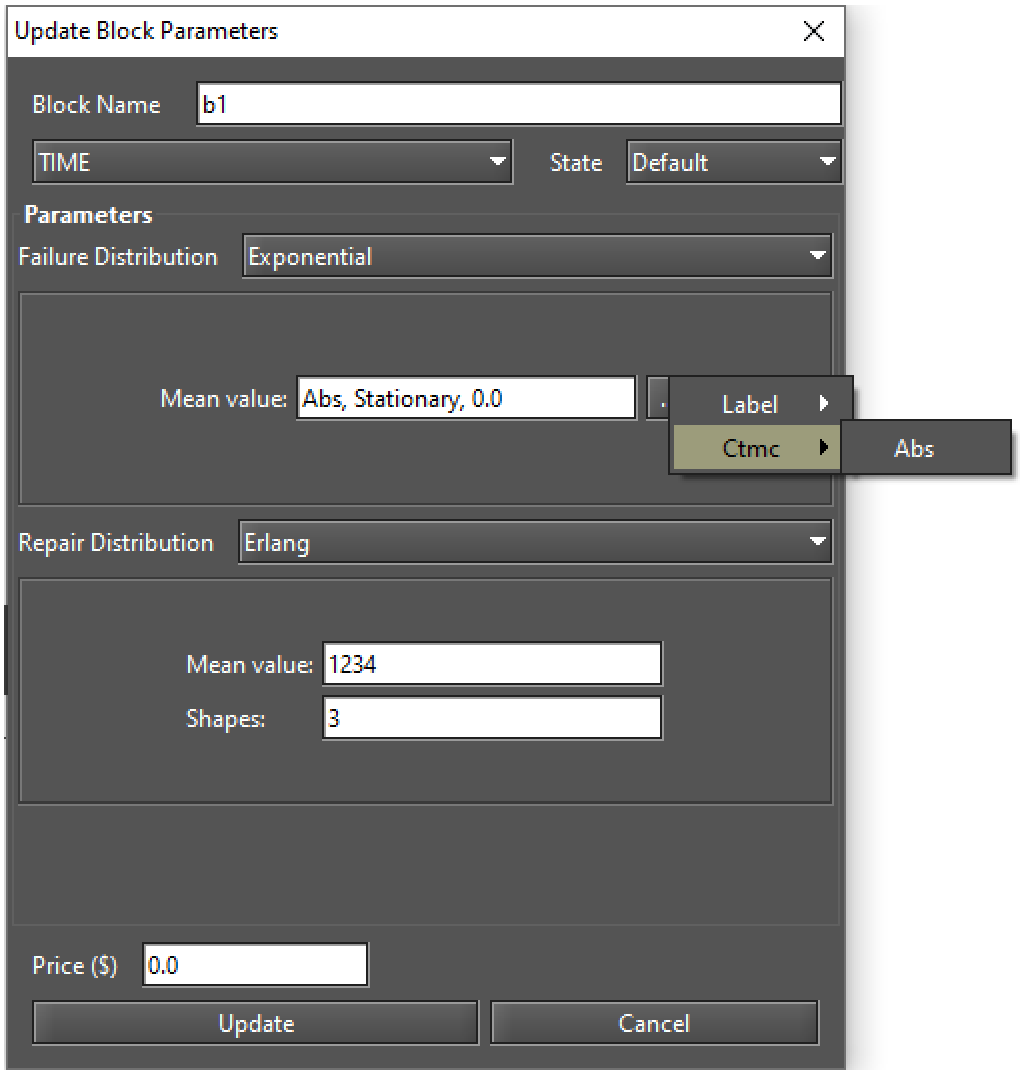
Mean value: 0.0

Price ($): 0.00

K: 1 N: 1

Figure 75: Inserting a k-out-of-n Block

Another way to edit the properties of a block is to right-click on it, and selecting the "Properties" menu item from the menu that appears. Figure 76 shows the properties of a simple block already created. When inserting a block, the "Block Name" field is not presented as the name of that block is defined by Mercury (see Figure 74). Once a block is created, the user may change its name by accessing its properties as we can see in the "Update Block Parameters" dialog depicted in Figure 76. Besides that, when inserting a block, the user has the option to define the number of blocks to be inserted. If more than one block is created, they will have the same parameter values entered by the user.


Update Block Parameters

Block Name: b1

TIME

State: Default

Parameters

Failure Distribution: Exponential

Mean value: Abs, Stationary, 0.0

Repair Distribution: Erlang

Mean value: 1234

Shapes: 3

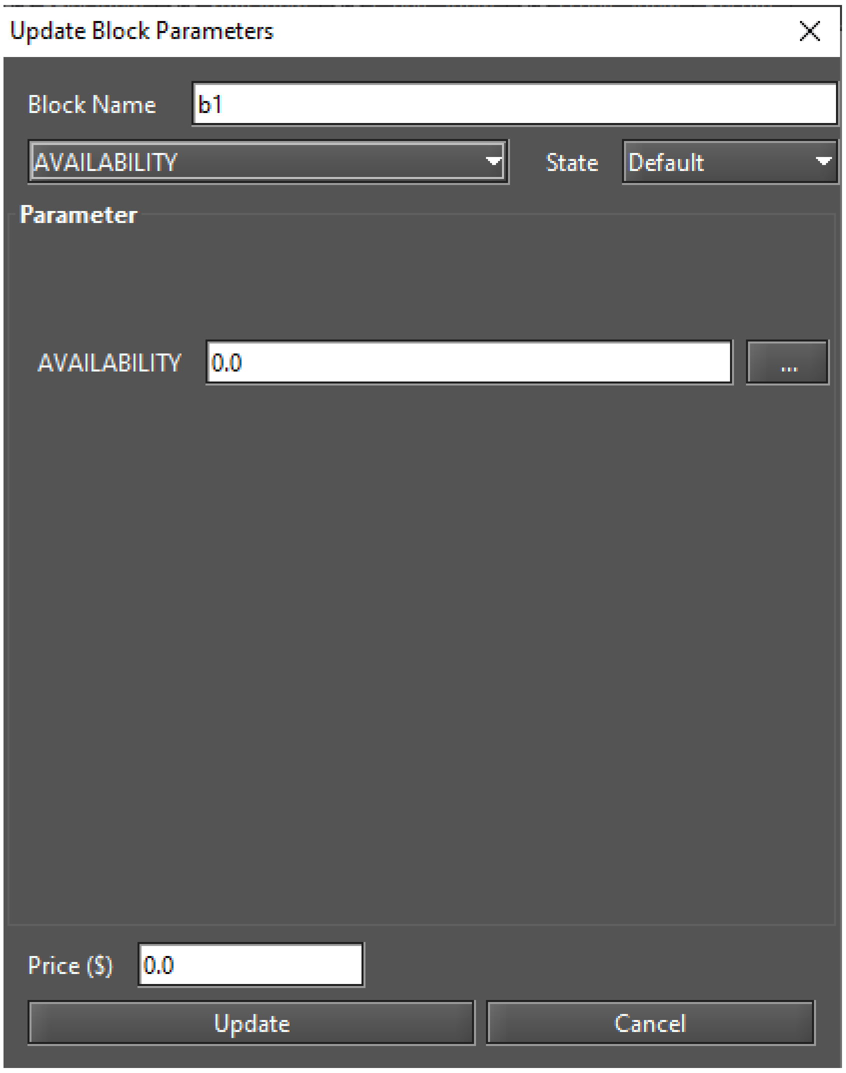
Price ($): 0.0

Figure 76: Simple Block Properties Dialog

Below, we describe each field of this dialog box

- **Number of Blocks.** When adding a new block, it is necessary to enter the number of blocks to be created having the same parameter values entered by the user. This field appears only when inserting a block.
- **Block Name.** Name of the block. The Mercury tool defines the name of the block when inserting it in the RBD for the first time, but it can be changed later.
- **Parameters Type.** There are four parameters types available to be chosen: RATES, TIME, AVAILABILITY, and RELIABILITY. The default parameter type is TIME.

  When the parameter type is TIME or RATES, the user can enter the values for the failure and repair parameters (see Figures 74, 75, and 76). On the other hand, when the type is AVAILABILITY or RELIABILITY, the user must enter the value corresponding to the selected parameter type, as we can see in Figure 77. In this example, the user must enter the availability of the component represented by the block.


Update Block Parameters

Block Name: b1

AVAILABILITY

State: Default

Parameter

AVAILABILITY: 0.0

Price ($): 0.0

Figure 77: Block Properties Dialog

- **State.** The state of the block. There are two states available to be chosen: DEFAULT or FAILED. The default state is DEFAULT. Thus, when having the state parameter as DEFAULT, it means that the block is working properly. On the other hand, having the state parameter as FAILED, it indicates that that block has failed.

- **Failure Parameters.** The Mercury tool supports a large number of probability distributions. Moreover, depending on the chosen distribution, the fields related to the parameters of that distribution appear in order for the user to enter their values. Besides that, the user may attach a label to the failure parameter or define this parameter by means of metrics computed from a CTMC model, which we call as hierarchical composition. See Section 3.1 for further information about hierarchical evaluations.

  The button **"..."** enables the user to select any label declared or metric defined in a CTMC model as hierarchical parameters from submodels.

- **Repair Parameters.** The fields related to the parameters of the chosen distribution appear for the user to enter their values. Besides that, the user may attach a label to the repair parameter or define this parameter by means of metrics computed from a CTMC model.

- **Price.** It refers to the cost regarding the component represented by the block. That cost is considered when evaluating the component importance and total cost of acquisition. For further information about this evaluation, see Section 3.3.4.

- **K and N.** When handling k-out-of-n blocks, two fields appears in the dialog box. A KooN block represents a set of identical components in a single block. It represents how many components at least (K) must be working for the failure does not occur, considering the (N)umber of components entered (K-out-of-N). Figure 78 shows a KooN block and the values for the parameters K and N are presented in the block description.

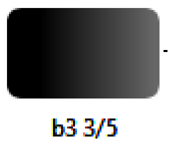


Figure 78: KooN Block

Figure 79 shows an RBD with two blocks in series and Figure 80 shows an RBD with two blocks in parallel. When all required block's parameters are entered, the block becomes dark. Evaluations in an RBD can only be carried out when all blocks are darkened.

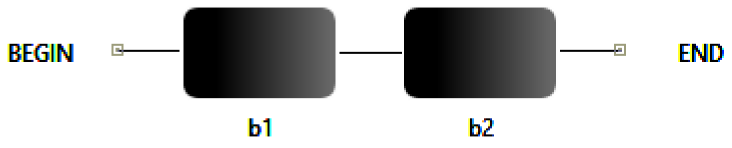


Figure 79: Two Blocks in Series

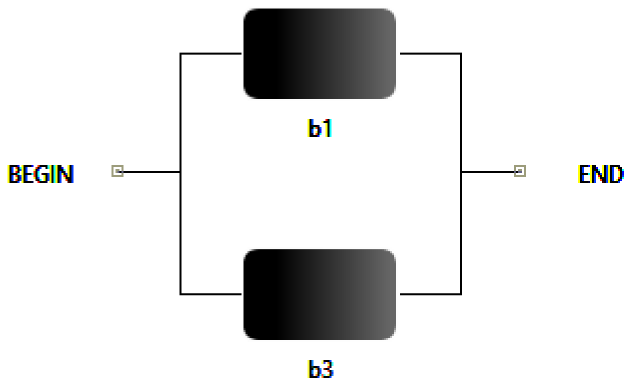


Figure 80: Two Blocks in Parallel

Now, let us see the types of blocks and how they are graphically represented. Figure 81 shows the four types of RBD blocks. Figures 81.a and 81.b demonstrate blocks with failure, repair, reliability or availability parameter assigned by the user, but in a) the state parameter is defined as “Default” while in b) it is defined as “Failed”. On the other hand, Figures 81.c and 81.d show blocks with no failure, repair, reliability, or availability parameter assigned, but in d) the state parameter is set as “Failed”.

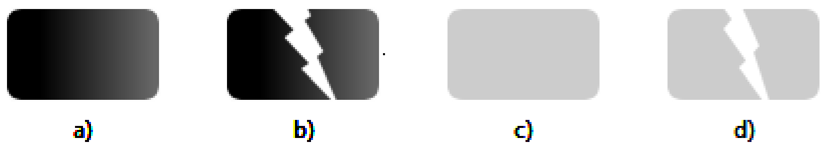


Figure 81: Types of RBD Blocks

3.1 Hierarchical Evaluation

RBD parameters can be set by evaluation of submodels such as CTMC. This kind of evaluation uses a metric result from a submodel as input to one or more RBD parameters. So, combinatorial and state-space models may be hierarchically combined in order to get the best of both models, e.g., an RBD could be used to represent the dependability relationship between independent subsystems, while more complex fail and repair mechanisms are modeled with CTMCs [4].

Mercury supports evaluation from a CTMC as a submodel to an RBD. At evaluation time, the CTMC will be solved and the chosen parameters will be gotten from that. Thus, it is necessary to have both CTMC and RBD models drawn inside of the same Mercury project. For example, let us assume that the following CMTC and RBD models exist in a project, as depicted in Figures 82 and 83, respectively.

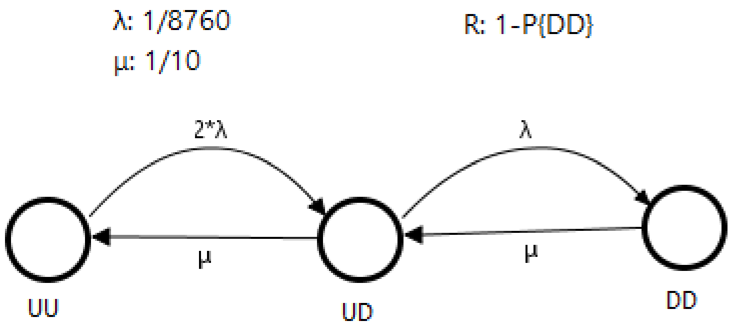


Figure 82: CTMC Submodel

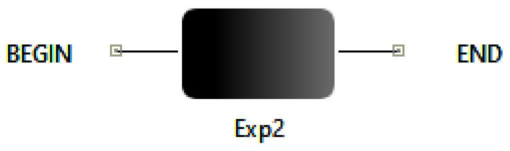
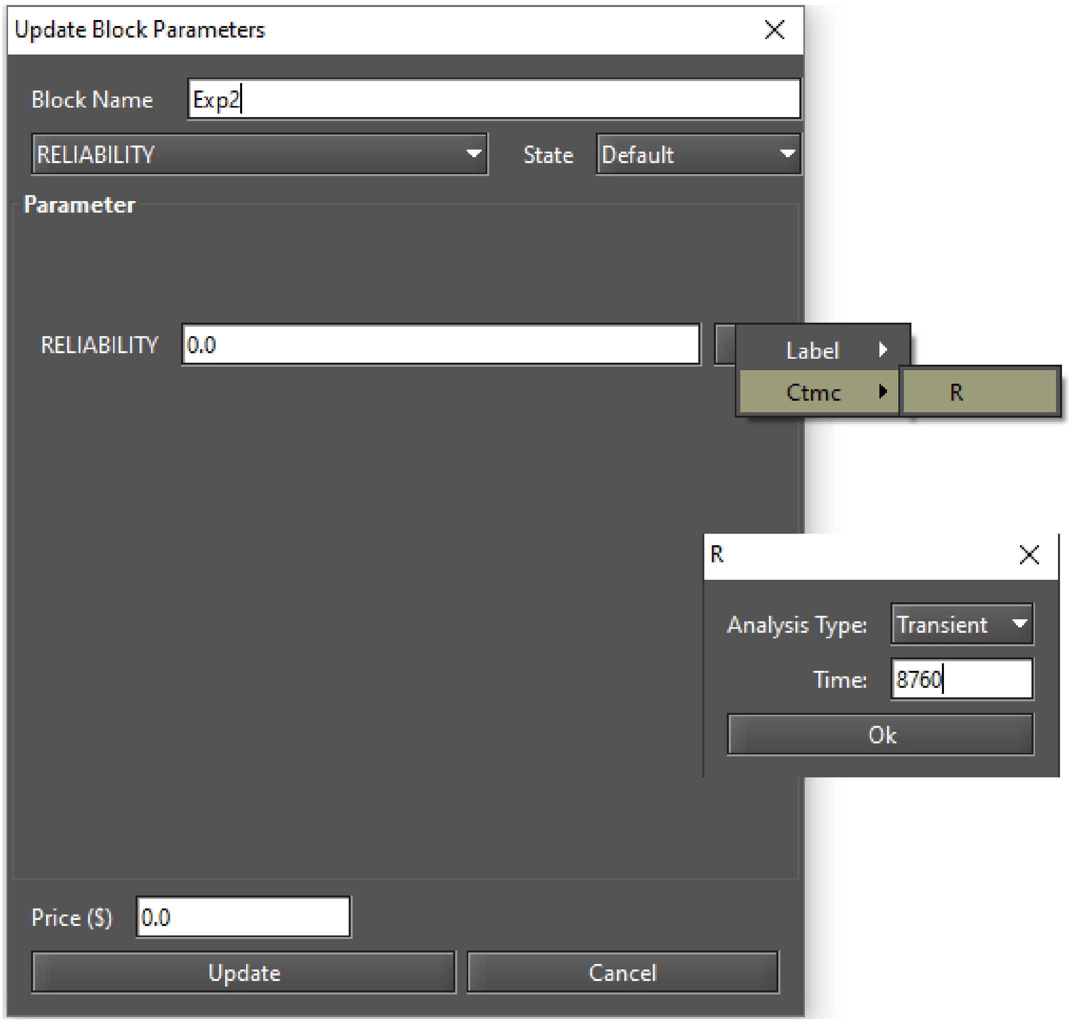


Figure 83: RBD Model

When the submodel is created, it is possible to get the parameter from it by clicking on the "..." button adjacent to the parameter, as we can see by looking at the Figure 84. Following, one can choose the desired metric from the submodel on the "Ctmc" submenu and then select the kind of analysis (stationary or transient). When the transient analysis is chosen, it is necessary to provide the time parameter.

In this example, we have shown the hierarchical evaluation for the reliability parameter, but it is also possible to get two parameters simultaneously from the submodel. That is the case when the parameters for MTTF (or failure rate) and MTTR (or repair rate) are exponentially distributed.



Block Name: Exp2
RELIABILITY
State: Default
RELIABILITY: 0.0
Analysis Type: Transient
Time: 8760
Price ($): 0.0

Figure 84: Hierarchical Evaluation for the Reliability Parameter

3.2 RBD Reduction

In order to reduce the RBD complexity, a feature is available aiming to reduce the number of blocks by a reduction process. This feature can be used until there is only one single block in the model. The RBD, for instance, may be reduced from its configuration such as series or parallel. The main objective when using this function is to simplify the model. However, the original parameters of the blocks may be lost and only metrics and characteristics of the original model will be preserved.

Figure 85 shows an RBD model before the reduction process is applied, in which there are four blocks. In order to perform the reduction, the user must right-click on the desired block and select the option "Apply Reduction" on the pop-up menu that appears (see Figure 86). Figure 87 illustrates the model after performing the reduction process, in which it is possible to see that the number of blocks was reduced to three (b2 and b3 were reduced to just one block, namely, ssb6). Thus, the result is a new reduced submodel.

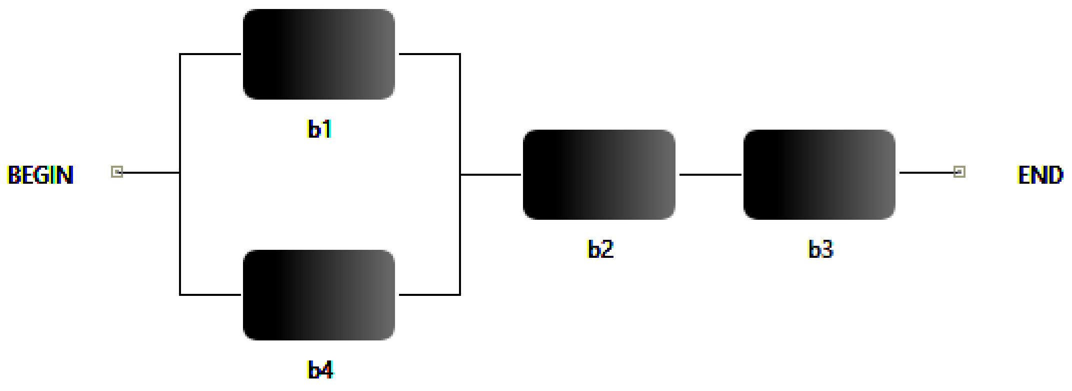


Figure 85: RBD before Reducing the Series Submodel

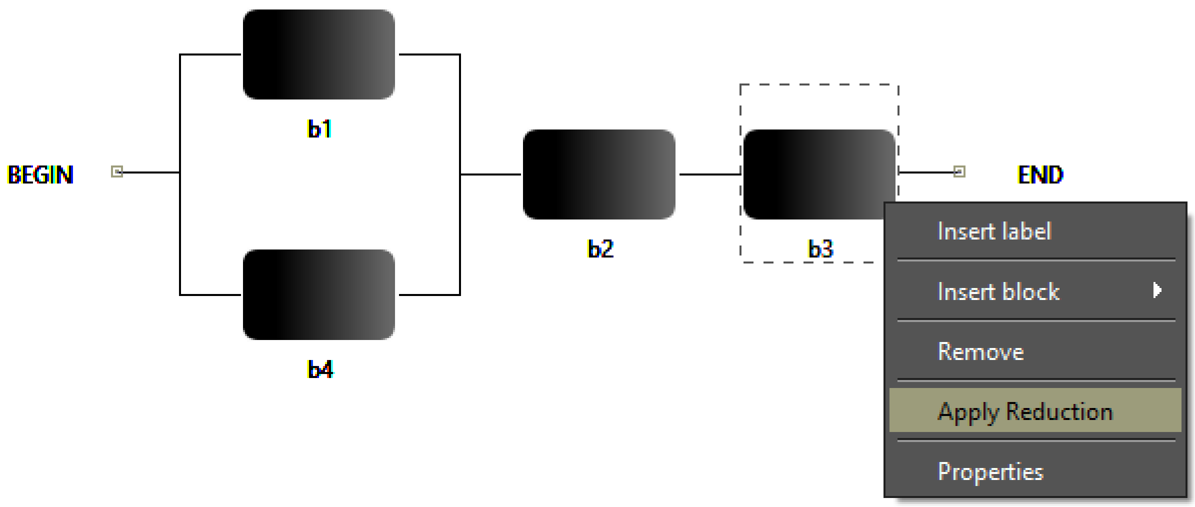


Figure 86: Applying Reduction in the RBD

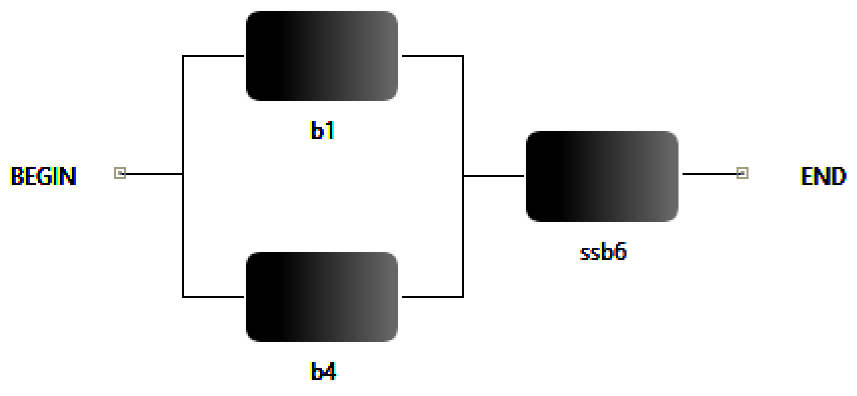
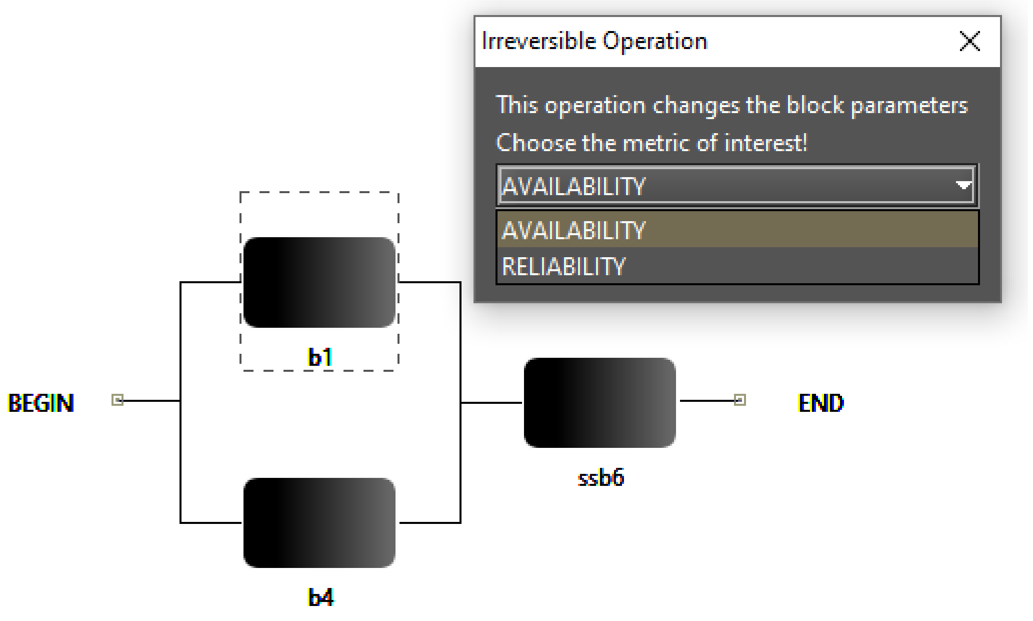


Figure 87: RBD after Reducing the Series Submodel

In a series sub-model, the original characteristics of the model are maintained. However, applying reductions in parallel sub-models, when the parameters of TIME or RATE have been edited, it is possible only to keep the characteristics at one point in time. So, when the reduction is applied in a parallel submodel, new options are displayed as depicted by Figure 88, allowing to choose the metric to be evaluated. If the reliability is selected, the time for reliability estimation is requested, as shown by Figure 89. Figure 90 shows the model after reducing the parallel submodel.


Irreversible Operation

This operation changes the block parameters
Choose the metric of interest!

Figure 88: RBD before Reducing the Parallel Submodel

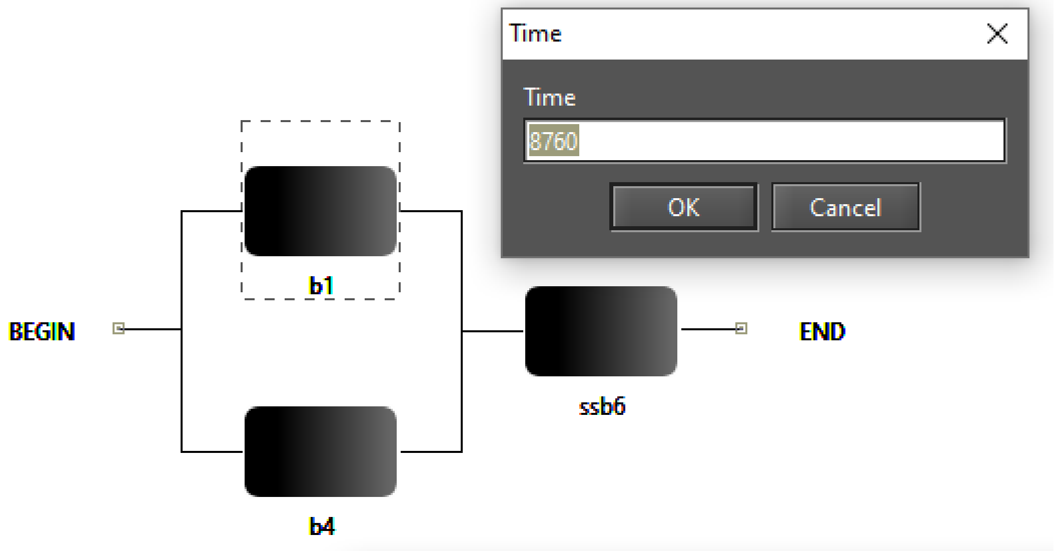


Figure 89: Time for Reliability Parameter Evaluation for Reducing the Parallel Submodel

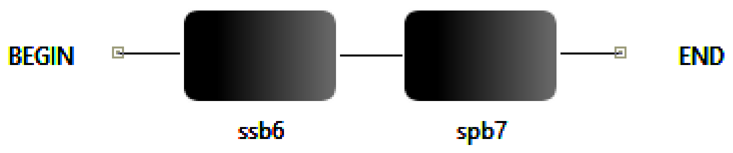


Figure 90: RBD after Reducing the Parallel Submodel

3.3 RBD Evaluation

The Mercury tool makes it possible to carry out a large number of evaluations on RBD models. The tool provides six functionalities for RBD evaluations: Exact Evaluation, Bounds Evaluation, Importance Measures, Experiment, Get Functions, and Sensitivity Analysis. All these options are available in the menu Evaluate -> RBD Evaluation as we can see in Figure 91. In the next subsections, we will introduce each one of these functionalities.

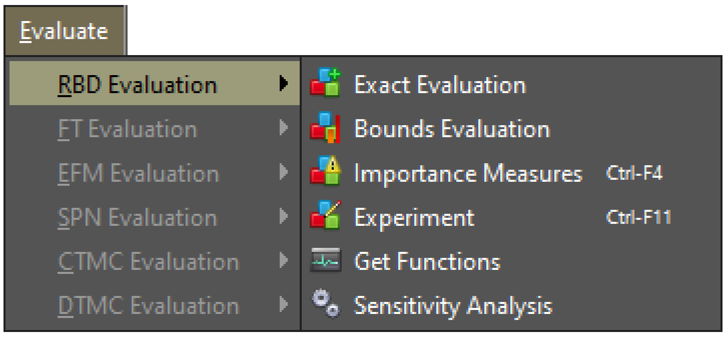
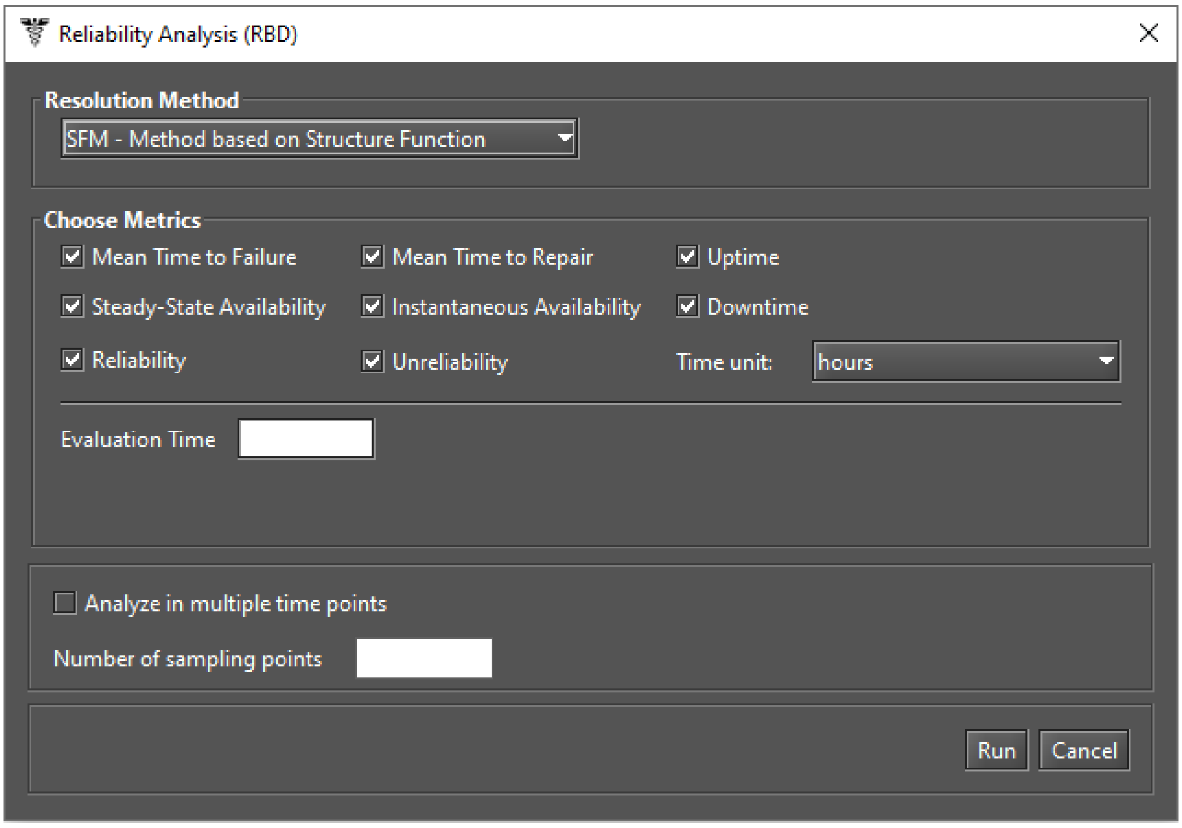


Figure 91: RBD Evaluation Menu

3.3.1 Exact Evaluation

The Exact Evaluation makes it possible to carry out a large number of dependability analyses on RBD models. This functionality is accessible in the menu Evaluate -> RBD Evaluation -> Exact Evaluation. Figure 92 shows the dialog box for dependability evaluation on RBDs.


Reliability Analysis (RBD)

Resolution Method: SFM - Method based on Structure Function

Choose Metrics

☑ Mean Time to Failure ☑ Mean Time to Repair ☑ Uptime

☑ Steady-State Availability ☑ Instantaneous Availability ☑ Downtime

☑ Reliability ☑ Unreliability Time unit: hours

Evaluation Time

☐ Analyze in multiple time points

Number of sampling points

Figure 92: Input window for RBD Analisys

As we can see, users can evaluate eight metrics. The metrics they can choose are Mean Time to Failure, Mean Time to Repair, Steady-State Availability, Instantaneous Availability, Reliability, Unreliability, Uptime, and Downtime. Also, it is possible to choose the time unit to be considered when computing Uptime and Downtime: seconds, minutes, hours, and days. When time-dependent metrics are chosen — Reliability, Unreliability, or Inst. Availability —-, users need to enter the time parameter — "Evaluation Time" field. Besides that, there is an option that analyze the selected time-dependent metrics by considering multiple points on time. The time interval from 0 to the evaluation time is divided by considering the number of points entered and, after that, the metric is computed for each point.

The Mercury tool provides two methods for computing dependability metrics. The user must choose between the SFM (Method based on Structure Function) or SDP (Sum of Disjoint Products) methods, as depicted in the Figure 93. SFM computes metrics by considering the structural function of the RBD model. On the other hand, the SDP method, based on Boolean algebra, computes metrics by considering the minimal cuts and minimum paths.

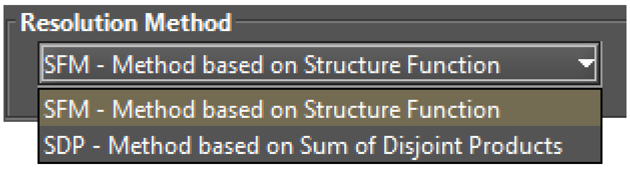
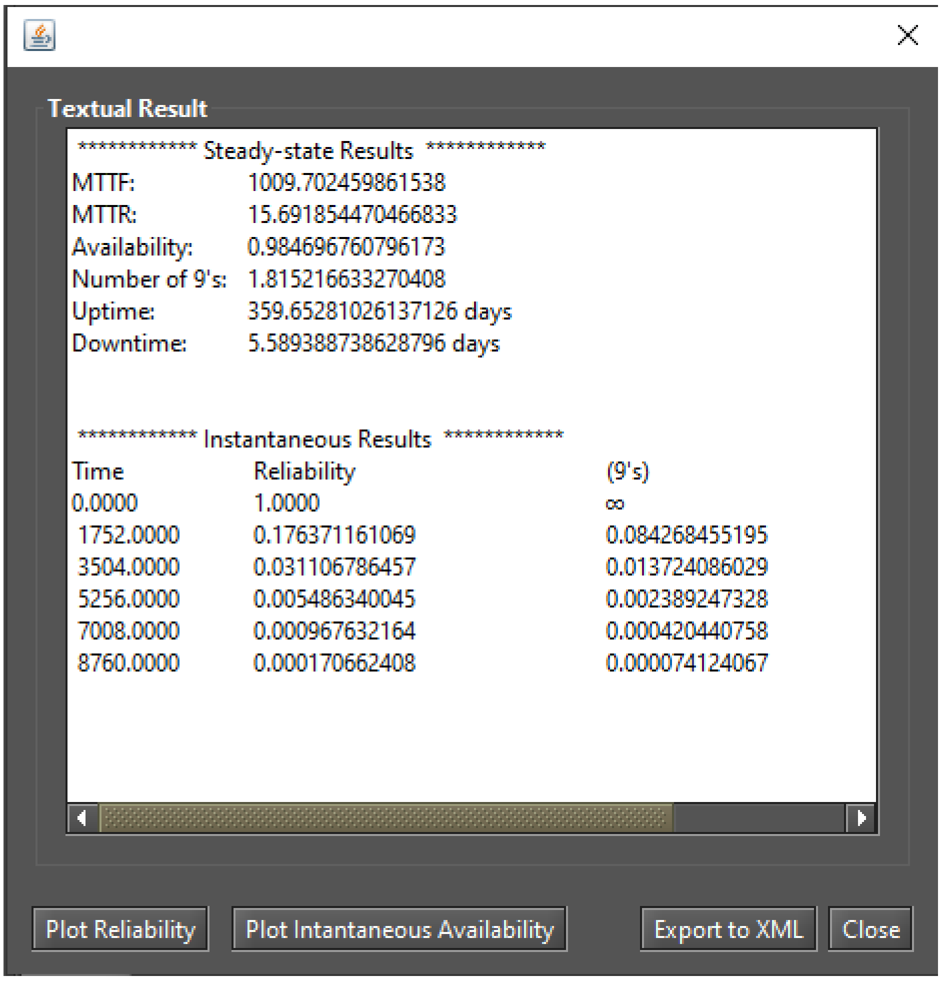

Resolution Method
SFM - Method based on Structure Function
SFM - Method based on Structure Function
SDP - Method based on Sum of Disjoint Products

Figure 93: Resolution Method

After choosing the desired options, and entered the evaluation time and the number of sampling points, if required, the user must click on the "Run" button in order to start the evaluation. At the end of the evaluation, a dialog box appears showing the results, as we can see in Figure 94. In this window, the results are divided into two groups. There exist the "Steady-state Results" for steady-state metrics and "Instantaneous Results" for time-dependent metrics. Listing 4 shows a result obtained by evaluating an RBD as an example.

Listing 4: Dependability Results for an RBD

```
************ Steady-state Results ************
MTTF:             1009.702459861538
MTTR:             15.691854470466833
Availability:     0.984696760796173
Number of 9's:    1.815216633270408
Uptime:           8631.667440364728 hours
Downtime:         134.14532963527165 hours


************ Instantaneous Results ************
Time      Reliability          (9's)               Unreliability
0         1.0000                                    0.0000
```



```
************ Steady-state Results  ************
MTTF:            1009.702459861538
MTTR:            15.691854470466833
Availability:    0.98469676079617 3
Number of 9's:   1.815216633270408
Uptime:          359.65281026137126 days
Downtime:        5.589388738628796 days


************ Instantaneous Results  ************
Time             Reliability                 (9's)
0.0000           1.0000                      ∞
 1752.0000       0.176371161069              0.084268455195
 3504.0000       0.031106786457              0.013724086029
 5256.0000       0.005486340045              0.002389247328
 7008.0000       0.000967632164              0.000420440758
 8760.0000       0.000170662408              0.000074124067
```

Figure 94: Dependabililty Results

1752	0.176371161069	0.084268455195	0.823628838931
3504	0.031106786457	0.013724086029	0.968893213543
5256	0.005486340045	0.002389247328	0.994513659955
7008	0.000967632164	0.000420440758	0.999032367836
8760	0.000170662408	0.000074124067	0.999829337592

Figure 95 shows the “Reliability Chart” dialog box displayed when clicking on the “Plot Reliability” button. Figure 96 shows the “Instantaneous Availability Chart” dialog box displayed when clicking on the “Plot Instantaneous Availability” button. These windows are only displayed when the reliability or instantaneous availability metric is selected in the input dialog box, respectively. The number of points on the plotted lines is determined by the number of points entered by the user.

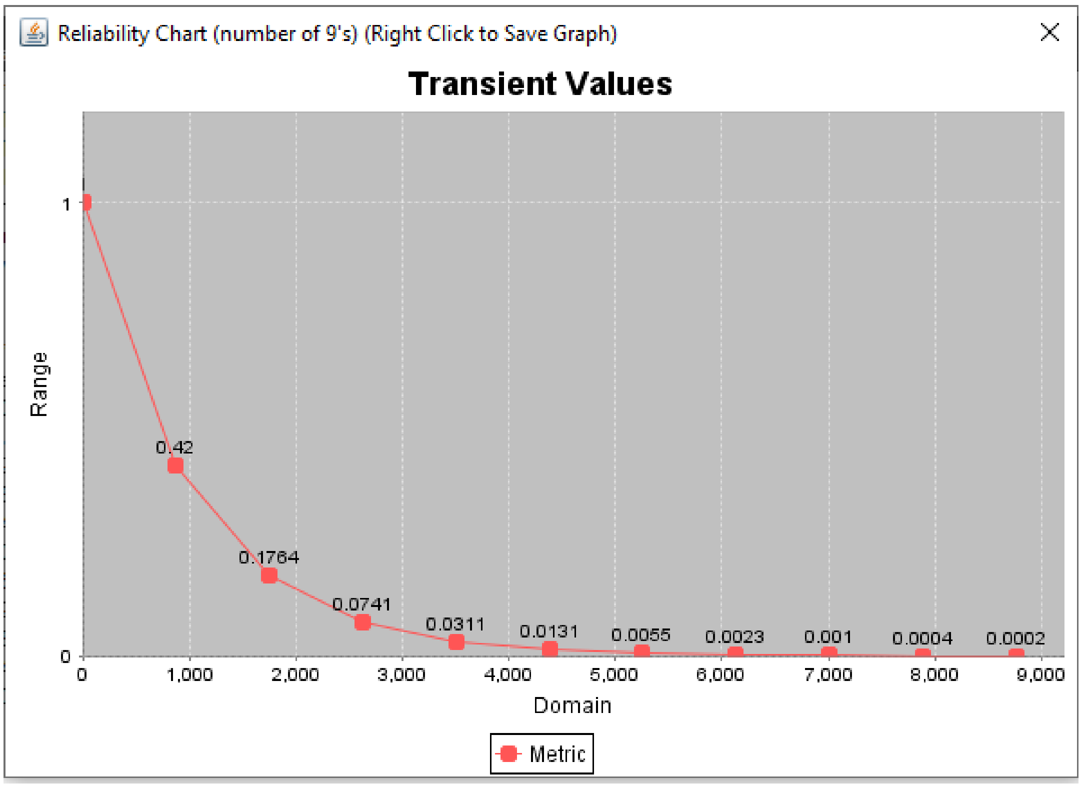


Figure 95: Plotting Reliability

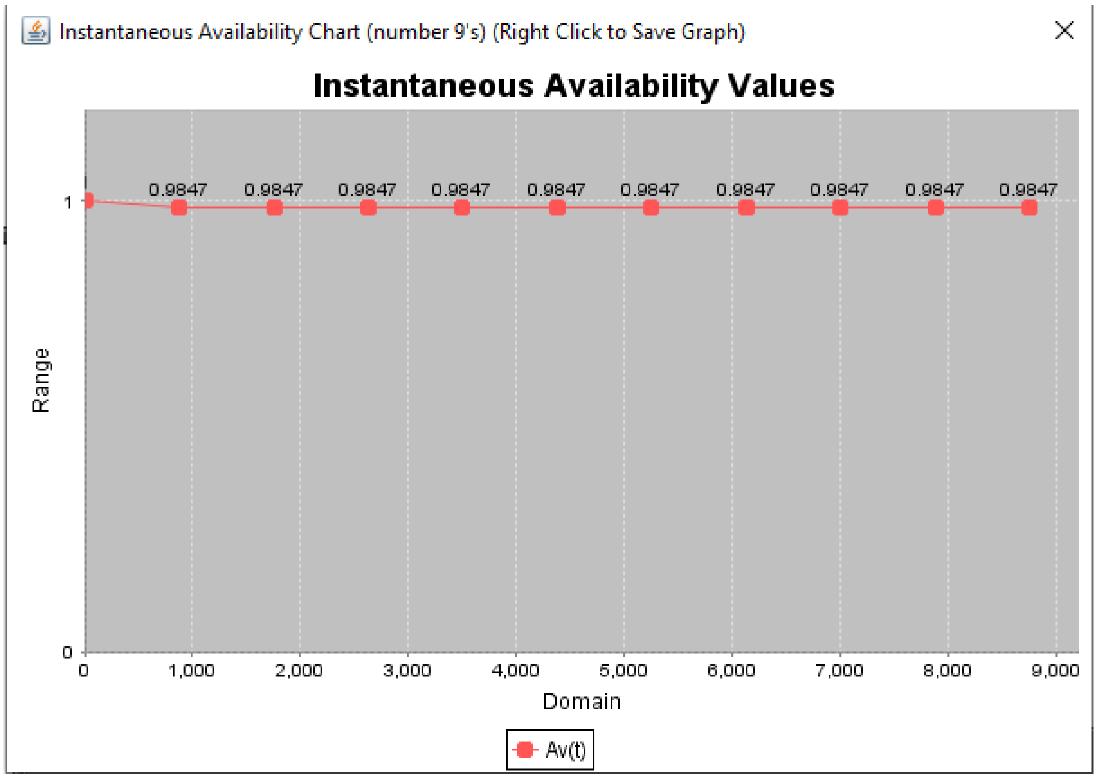
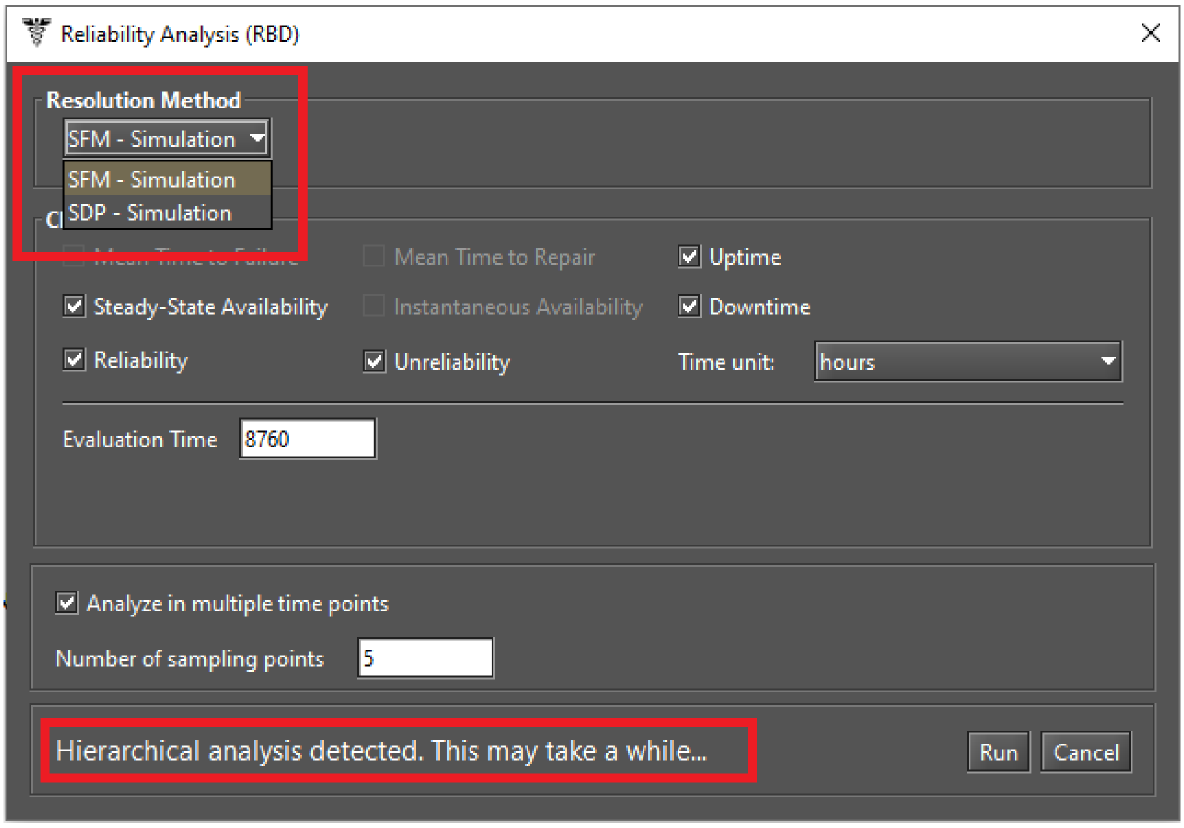


Figure 96: Plotting Instantaneous Availability

The resolution of an RBD through simulation occurs when the probability distribution regarding failure and/or repair of one or more components is non-exponential, such as Erlang, Gamma, among others. When this occurs, non-exponential blocks are converted into SPNs and then they are solved by simulation. When the Mercury tool detects this, it shows a message informing that hierarchical analysis has been detected, as we can see by looking at the Figure 97. Now, the tool makes available two resolution methods to be chosen: **SFM - Simulation** and **SDP - Simulation**.


Reliability Analysis (RBD)

Resolution Method: SFM - Simulation (options: SFM - Simulation, SDP - Simulation)

Mean Time to Failure, Mean Time to Repair, Uptime [✓], Steady-State Availability [✓], Instantaneous Availability, Downtime [✓], Reliability [✓], Unreliability [✓]

Time unit: hours

Evaluation Time: 8760

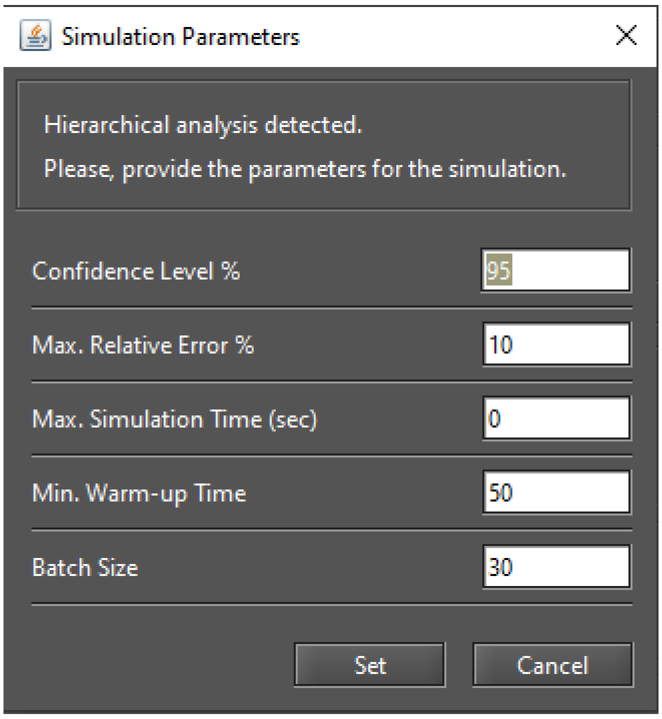
Analyze in multiple time points [✓]

Number of sampling points: 5

Hierarchical analysis detected. This may take a while...

Figure 97: Input window for RBD Hierarchical Analysis

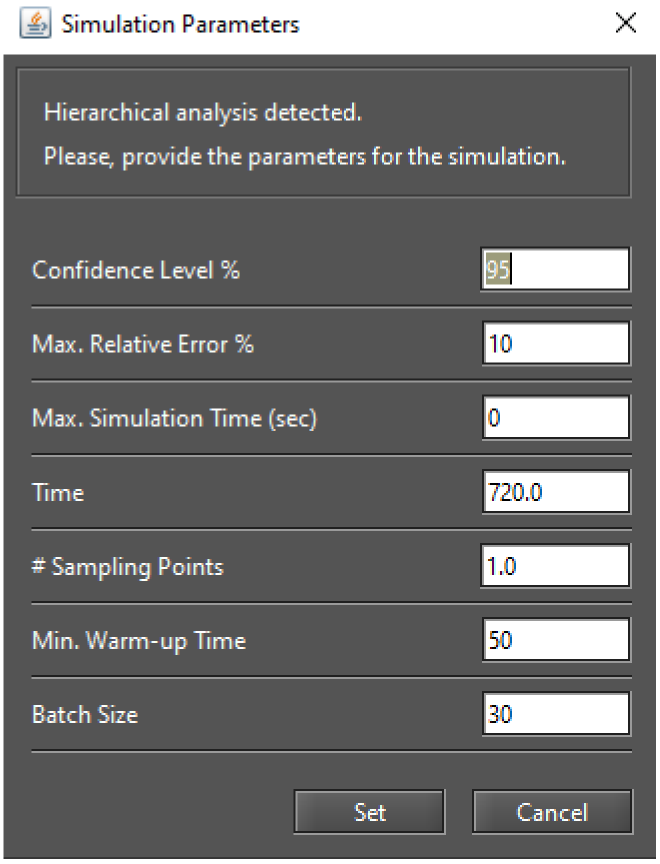
When clicking on the “Run” button, the user is prompted to provide some input parameters for simulation, according to the chosen metrics. When only steady-state metrics are selected, the dialog box presented by Figure 98 is shown. On the other hand, when one or more time-dependent metric is selected, the dialog box presented by Figure 99 is shown. After entering the required parameters and starting the simulation, the results are presented. It is important to highlight that when solving RBDs by simulation, it is not possible to solve the following metrics: mean time to failure, mean time to repair, and instantaneous availability. For further information about the input parameters for simulation, see Section 2.1.


Simulation Parameters

Hierarchical analysis detected.
Please, provide the parameters for the simulation.

Confidence Level %	95
Max. Relative Error %	10
Max. Simulation Time (sec)	0
Min. Warm-up Time	50
Batch Size	30

Figure 98: RBD Simulation for Steady-State Metrics


Simulation Parameters

Hierarchical analysis detected.
Please, provide the parameters for the simulation.

Confidence Level %	95
Max. Relative Error %	10
Max. Simulation Time (sec)	0
Time	720.0
# Sampling Points	1.0
Min. Warm-up Time	50
Batch Size	30

Figure 99: RBD Simulation for Time-Dependent Metrics

3.3.2 RBD Experiment

Sometimes, it is necessary to evaluate the impact of varying some parameters on the required metrics. The Mercury tool provides a feature for supporting this type of evaluation on RBDs. Now, we will demonstrate an example of running an experiment on an RBD model.

The first step for that aim is to define one or more labels. Labels are variables that store numeric values and they can be attached to failure/repair parameters of blocks. The value of a label is changed by considering the data entered by the user and, in each changing, the selected metric is evaluated.

To insert a label, the user must right-click on an RBD block and on the pop-up menu that appers the user must choose the “Insert label” menu item, as depicted by Figure 100.

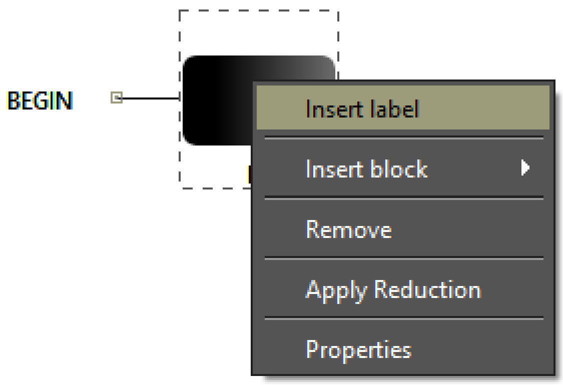
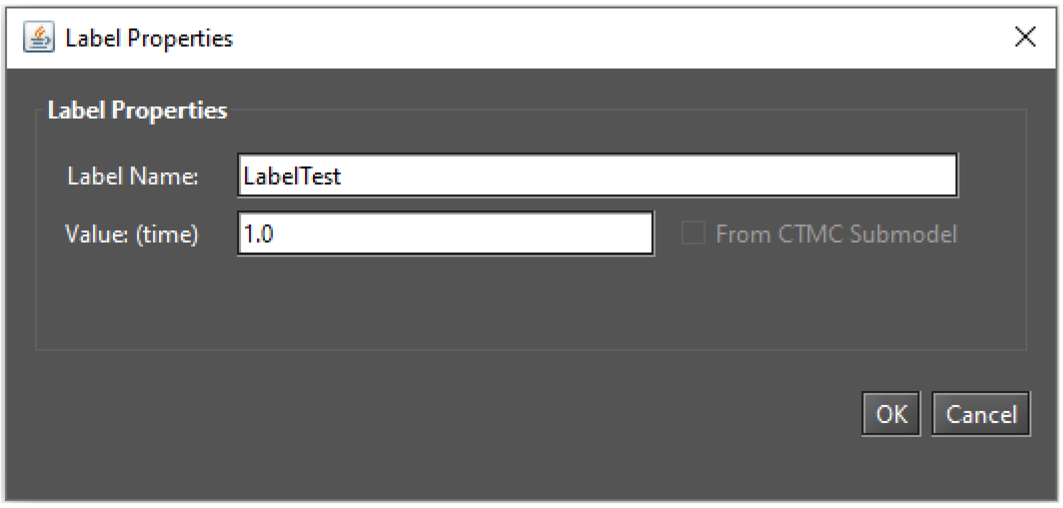


Figure 100: Insert Label Menu

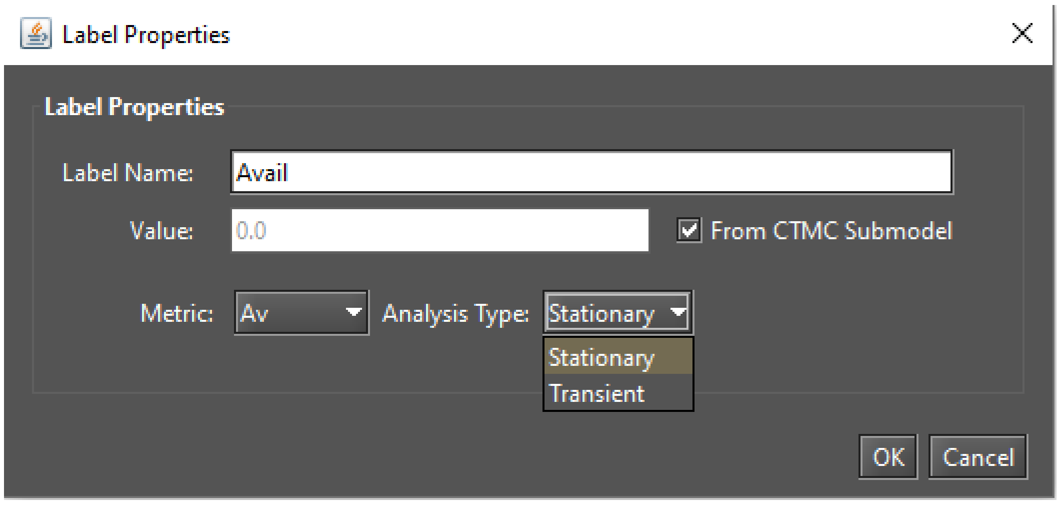
After that, the “Label Properties” input dialog box is shown (see Figure 101). In this dialog box, the user must set the properties of the label.



Label Name: LabelTest
Value: (time) 1.0
From CTMC Submodel

Figure 101: RBD Label Properties

The user can define the failure/repair parameter of blocks by means of metrics computed from a CTMC model, which we call as hierarchical composition. This is done by assigning a CTMC metric to a label, as shown in Figure 102. The user must check the box “From CTMC submodel", and select one of the metrics previously defined in the CTMC. The type of analysis (Stationary or Transient) must be selected too. In the case of transient analysis, the time parameter is also required.


Label Properties

Label Properties

Label Name: Avail

Value: 0.0 [✓] From CTMC Submodel

Metric: Av Analysis Type: Stationary

Figure 102: Assigning a CTMC Metric to an RBD Label

When a label is inserted, it is available in the left-side panel of the RBD tab (see Figure 103).

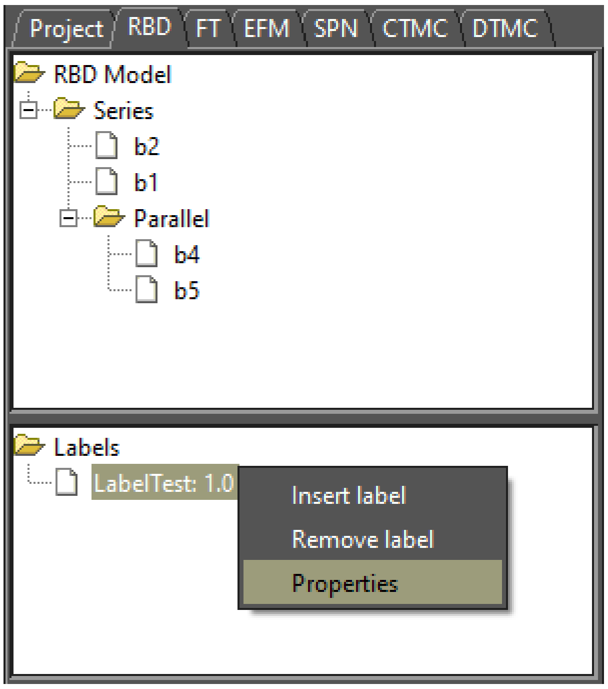
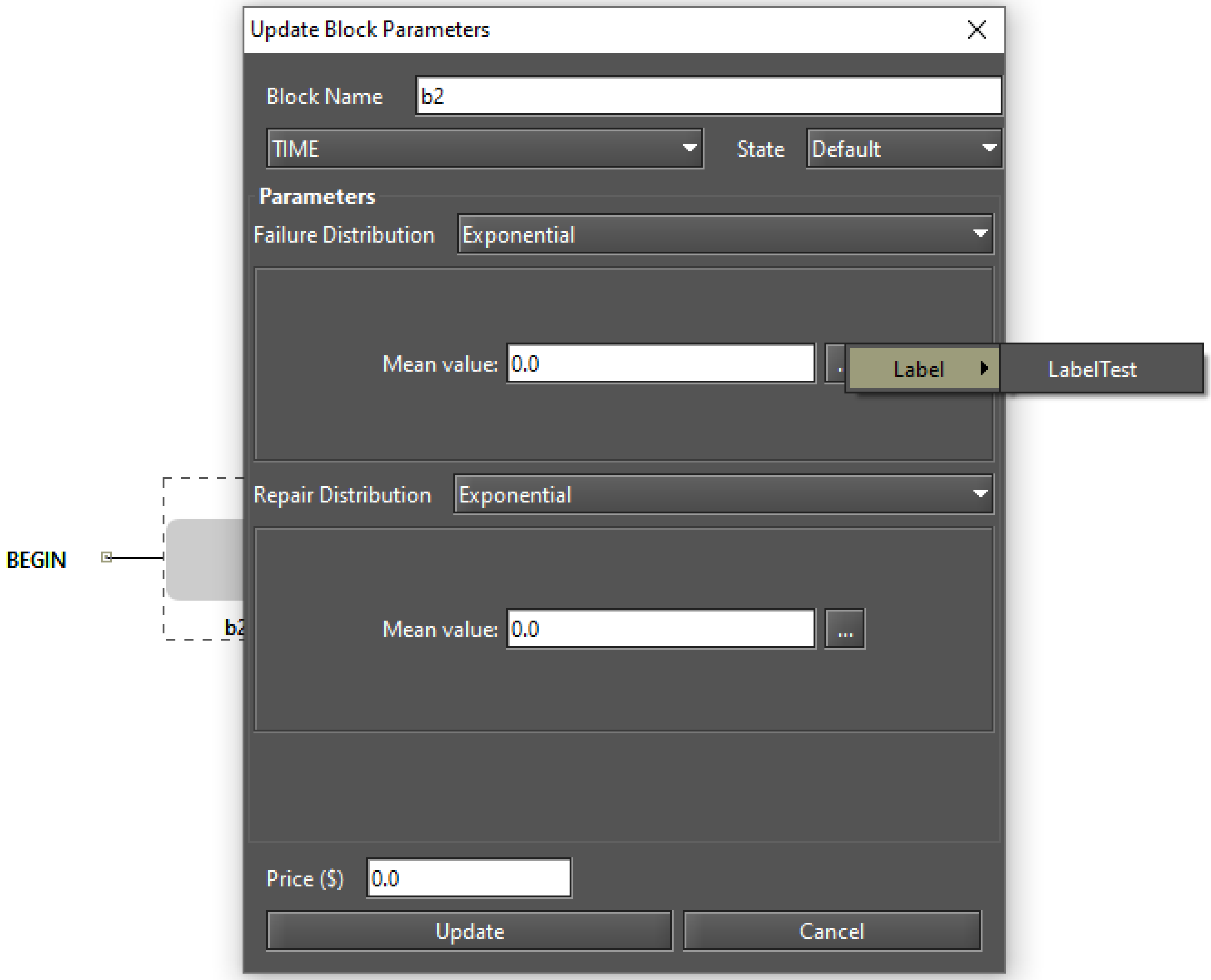


Figure 103: RBD Left-Side Panel

By right-clicking on any inserted label, a pop-up menu with three menu items appears, as we can see by looking at the last figure. Below, we describe each of them.

- **Insert label.** Show the “Label Properties” dialog box that allows inserting a new label.
- **Remove label.** Remove the selected label.
- **Properties.** Show the “Label Properties” dialog box that allows changing the properties of the label.

After defining a label, it is necessary to attach this label to the failure/repair parameter of the block under evaluation. Figure 104 demonstrates how to attach a label to a failure/repair parameter of a block.


Update Block Parameters

Block Name: b2

TIME

State: Default

Parameters

Failure Distribution: Exponential

Mean value: 0.0

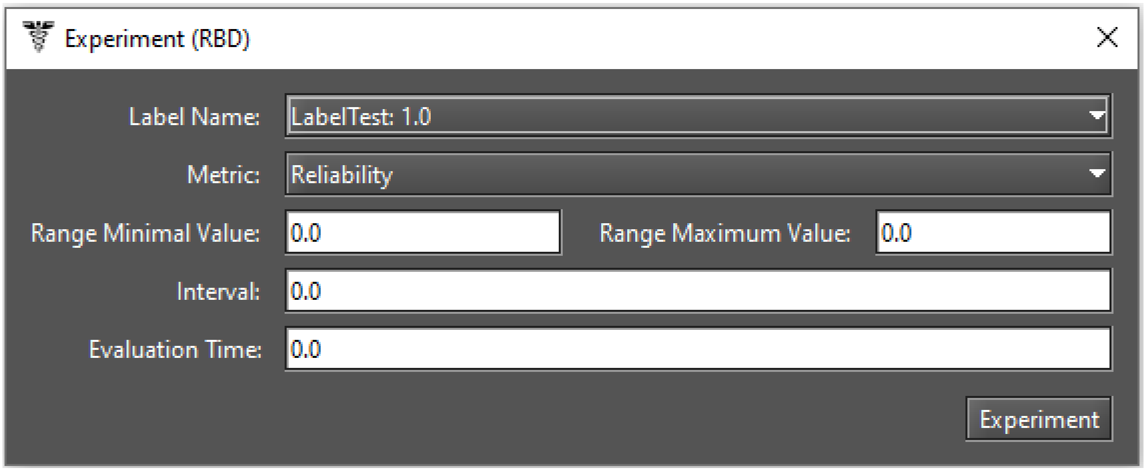
Repair Distribution: Exponential

Mean value: 0.0

Price ($): 0.0

Figure 104: Attaching a Label to a Block Parameter

The feature for RBD experiment is accessible in the menu Evaluate -> RBD Evaluation -> Experiment. Figure 105 shows the “RBD Experiment” dialog. In order to perform the experiment, the user must fill the required fields.


Experiment (RBD)

Label Name: LabelTest: 1.0
Metric: Reliability
Range Minimal Value: 0.0
Range Maximum Value: 0.0
Interval: 0.0
Evaluation Time: 0.0

Figure 105: RBD Experiment Dialog

Below, we describe each of them.

- **Label Name.** The label to have its value changed.
- **Metric.** The metric to be evaluated.
- **Minimum Value.** The initial value to be assigned to the selected label.
- **Maximum Value.** The final value to be assigned to the selected label.
- **Interval.** It is the step-size for changing the value of the label. The label starts with the minimum value and its value is increased by considering the entered interval. At each change, the selected metric is evaluated. The experiment ends when the maximum value for the label is reached.
- **Evaluation Time.** The evaluation time to be considered. For time-dependent metrics — reliability, unreliability, instantaneous availability — the user must enter the time parameter.

After defining the input parameters for the experiment, the user must click on the “Experiment” button in order to start the experiment. After that, when the experiment has been finished, the “Experiment Result” dialog is shown as depicted by Figure 106.

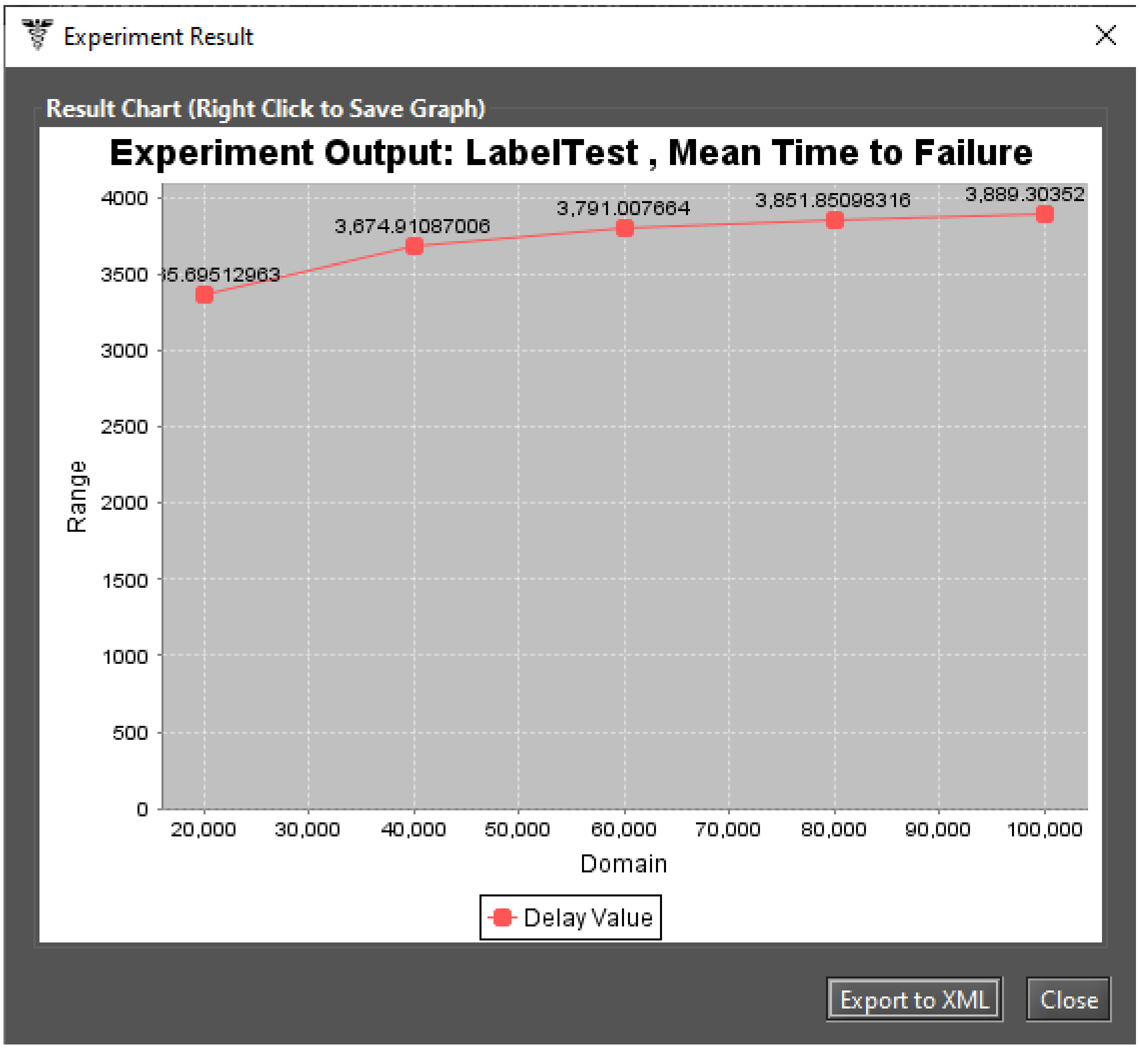


Figure 106: Experiment Result Dialog

3.3.3 Bounds for Dependability Analisys

Bounds for Dependability Analysis is adopted to estimate the dependability metrics through the calculation of reliability, availability, or downtime. Therefore, it is necessary to estimate the bounds values (upper and lower limits) for computing this analysis in which users quickly obtain the results. This functionality should be performed when the model in analysis is very large. This method is subdivided into two parts: (i) computing the bounds values and (ii) adopting the sum of disjoint points to find the successive values and the number of iterations.

It is accessible in the menu Evaluate -> RBD Evaluation -> Bounds Evaluation. Figure 107 shows the “Bounds for Dependability Analysis” dialog box. As we can see in the Figure 108, the user can evaluate four metrics: Steady State Availability, Instantaneous Availability, Reliability, and Downtime. The user needs to enter the time parameter for time-dependent metrics. After selecting the desired metric and entering the time, if required, the user needs to click on the button “Get Start Values” in order to start the evaluation.

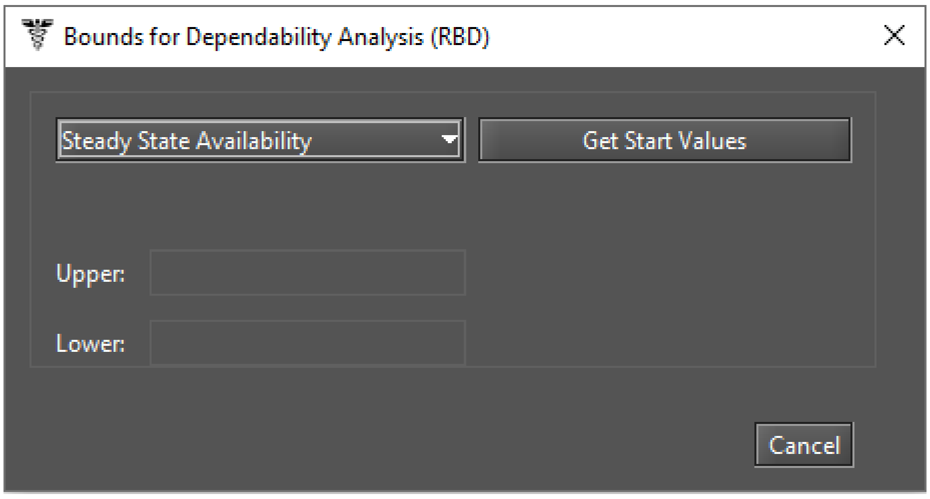

Bounds for Dependability Analysis (RBD)
Steady State Availability
Get Start Values
Upper:
Lower:
Cancel

Figure 107: Bounds for Dependability Analysis

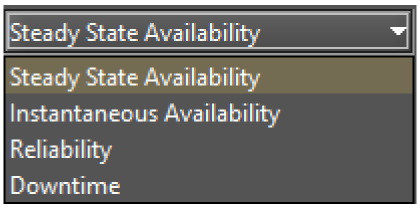
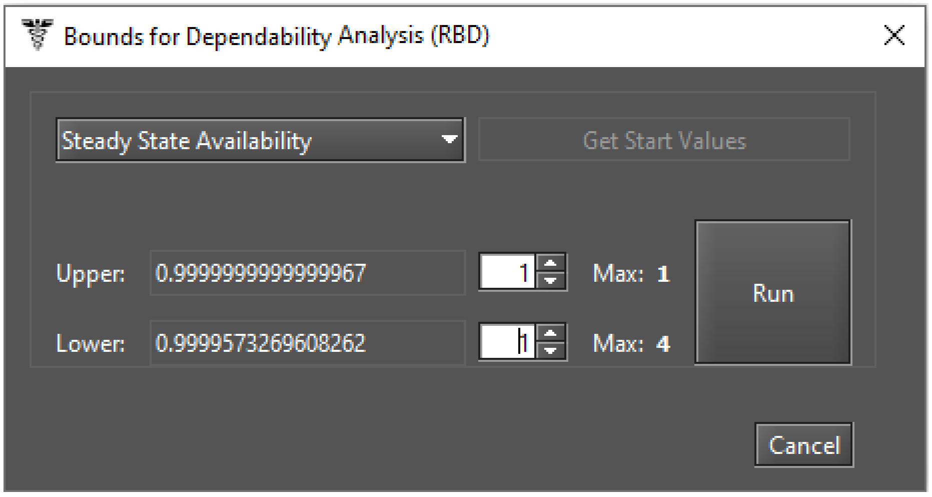

Steady State Availability
Steady State Availability
Instantaneous Availability
Reliability
Downtime

Figure 108: Metrics for Bounds Evaluation

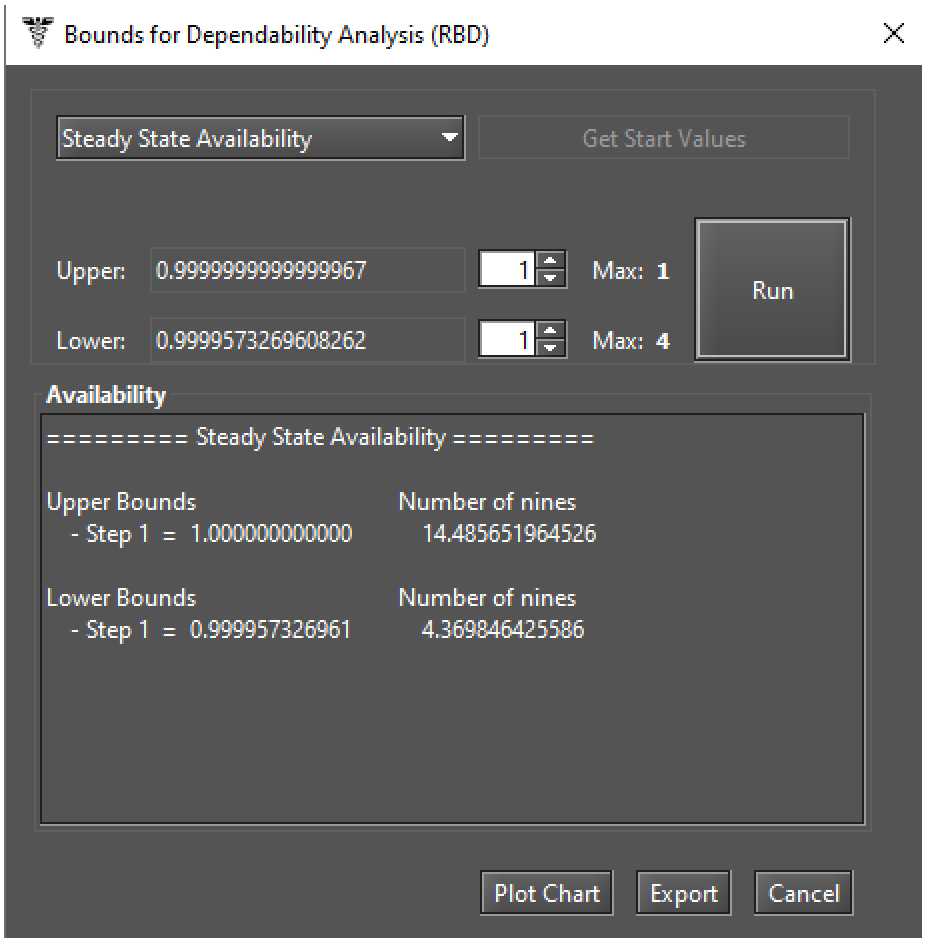
First, it is necessary to compute the upper and lower values. This option adopts the first path and the first cut to provide the higher and lower values of the chosen metric. The paths are related to the lower bounds, in which a minimal set of components is chosen to guarantee the operational mode of the system. Meanwhile, the cuts are related to the upper bounds, in which a minimal set of components that ensure the system on the failure mode is adopted. After getting the upper and lower values, the user can define the number of steps for the upper and lower values and click on the “Run” button (see Figure 109). Then, the user can see the result as demonstrated by Figure 110. It is possible to plot a chart and export the results to an MS Excel file.


Bounds for Dependability Analysis (RBD)
Steady State Availability
Get Start Values
Run
Cancel

Upper: 0.9999999999999967 1 Max: 1

Lower: 0.9999573269608262 1 Max: 4

Figure 109: Bounds for Steady State Availability Metric


Bounds for Dependability Analysis (RBD)
Steady State Availability
Get Start Values
Run
Availability
Plot Chart
Export
Cancel

Upper: 0.9999999999999967 1 Max: 1

Lower: 0.9999573269608262 1 Max: 4

```
========= Steady State Availability =========

Upper Bounds                  Number of nines
  - Step 1  =  1.000000000000   14.485651964526

Lower Bounds                  Number of nines
  - Step 1  =  0.999957326961   4.369846425586
```

Figure 110: Bounds for Steady-State Availability Metric - Steps Result

The method adopted to find successive values and the number of iterations (values) is defined by the number of paths and cuts of the model. Increasing the number of iterations, the value found will be closer to the exact value. When the calculation is done for the last path or last cut, the exact value for the metric is found. The exact values will be the value found on the last step.

3.3.4 Component Importance and Total Cost of Acquisition

Component importance is a metric that indicates the impact of a particular component in the system's overall. With these importance values, the most important component (i.e., that one with the highest importance) should be improved to increase system reliability or availability. This evaluation can assist in maintenance actions.

The Component Importance Measures functionality is responsible for identifying the relative importance of each component with respect to the overall system reliability or availability. To use the feature, users should select the "Importance Measures" in the menu Evaluate -> RBD Evaluation. After that, they should select one metric of interest on the "Component Importance Measures" dialog box and click on the "Evaluate" button (see Figure 111). If the parameter "Cost" was entered in the blocks' properties, it is possible to evaluate the relationship between these metrics and investment costs. For reliability metrics, it is necessary to enter the time parameter.

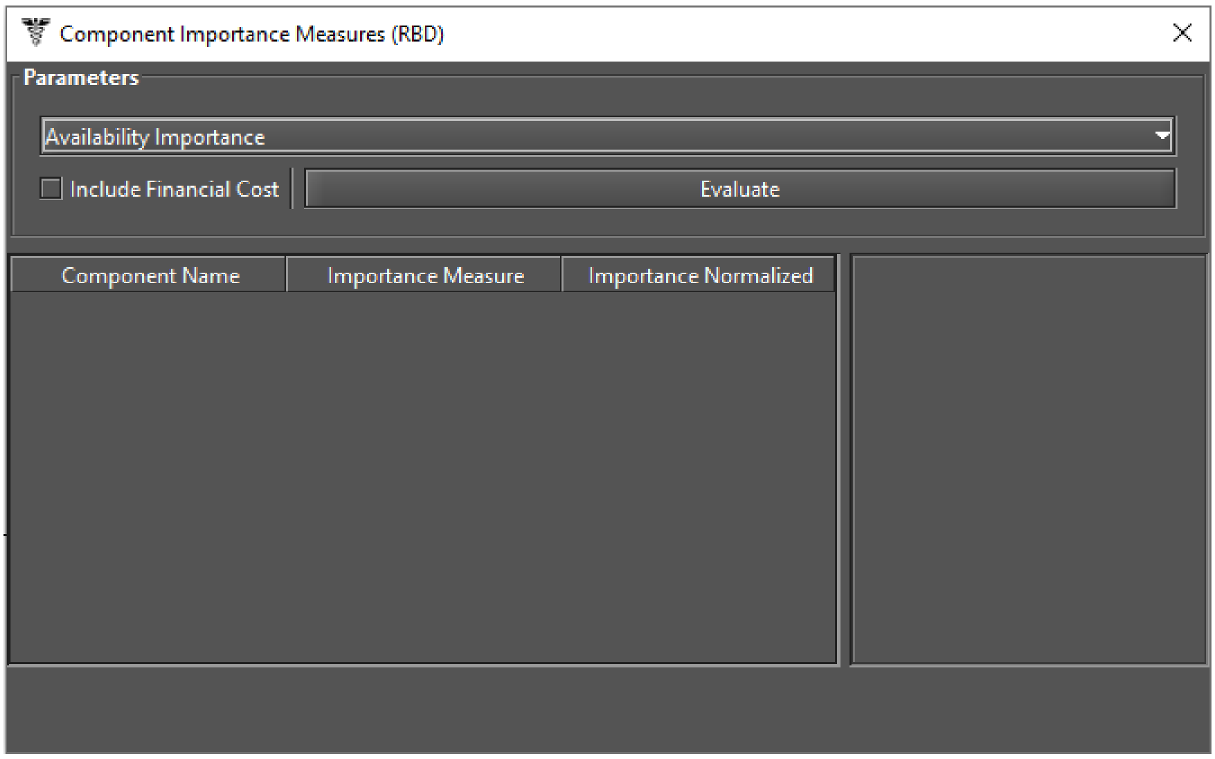


Figure 111: Component Importance Measures

It is possible to compute some importance measures as shown in Figure 112. The measures available are Availability Importance, Reliability Importance (Birnbaum), and Criticallity (Reliability or Availability) Importance (CRI). Last measures can be obtained considering the system's failure "(f)" or functioning.

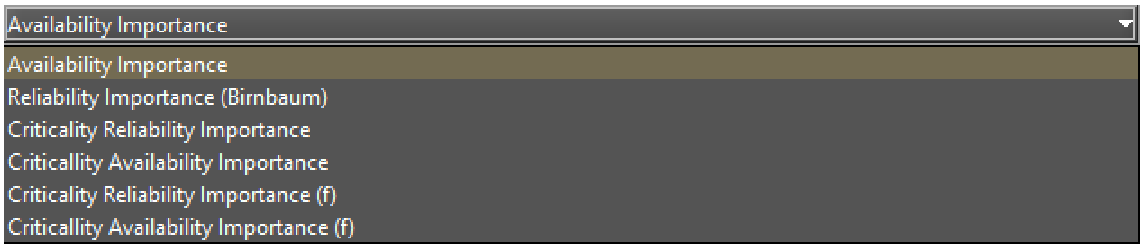
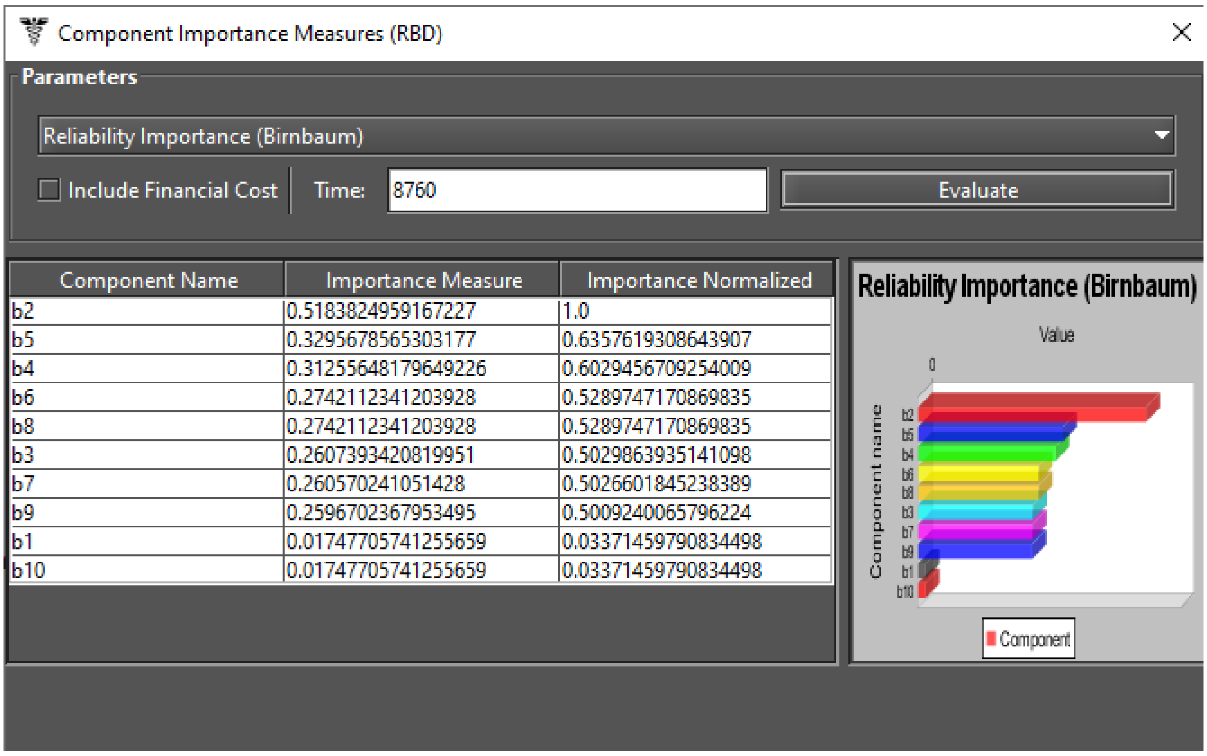

Availability Importance
Availability Importance
Reliability Importance (Birnbaum)
Criticality Reliability Importance
Criticallity Availability Importance
Criticality Reliability Importance (f)
Criticallity Availability Importance (f)

Figure 112: Component Importance Measures - Metrics of Interest

The results of this evaluation are shown by Figure 113. The results have shown the importance value for each component and a graphical view as a ranking highlighting those most significant in the analysis.


Component Importance Measures (RBD)
Parameters
Reliability Importance (Birnbaum)
Include Financial Cost
Time:
8760
Evaluate
Reliability Importance (Birnbaum)
Value
0
Component name
b2
b5
b4
b6
b8
b3
b7
b9
b1
b10
Component

Component Name	Importance Measure	Importance Normalized
b2	0.5183824959167227	1.0
b5	0.3295678565303177	0.6357619308643907
b4	0.31255648179649226	0.6029456709254009
b6	0.2742112341203928	0.5289747170869835
b8	0.2742112341203928	0.5289747170869835
b3	0.2607393420819951	0.5029863935141098
b7	0.260570241051428	0.5026601845238389
b9	0.2596702367953495	0.5009240065796224
b1	0.01747705741255659	0.03371459790834498
b10	0.01747705741255659	0.03371459790834498

Figure 113: Reliability Importance Results

3.3.5 Structural and Logical Functions

Mercury tool generates the structural and logic functions of the RBD models. Both functions are a behavioral representation of the system represented by the model. By applying them, it is possible to evaluate the impact of the failed components on the system status. Structural and logical functions are functions related to the states of the components. The system and each component can be in the working (default) or failed state. The system state is a binary random variable that is determined by the states of the components. If the states of the components are known, the state of the system is also known.

The state can be toggled via the "Properties" dialog box (see Figure 114) and when the state of the block is failed, the component is depicted by a broken icon, as we have aforementioned.

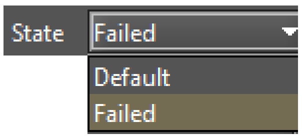


Figure 114: Block State

Now, let us demonstrate how to get these functions by using the Mercury tool. Figure 115 shows an RBD model with blocks in series and in parallel. As we can see, there is a failed block in this RBD (block b5).

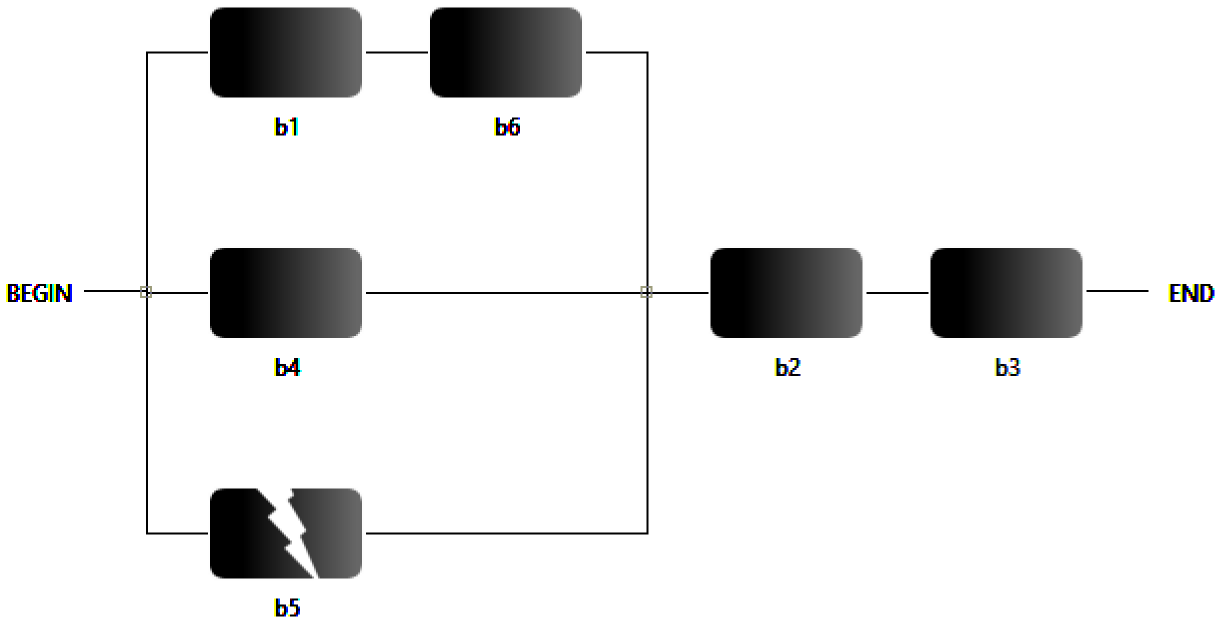
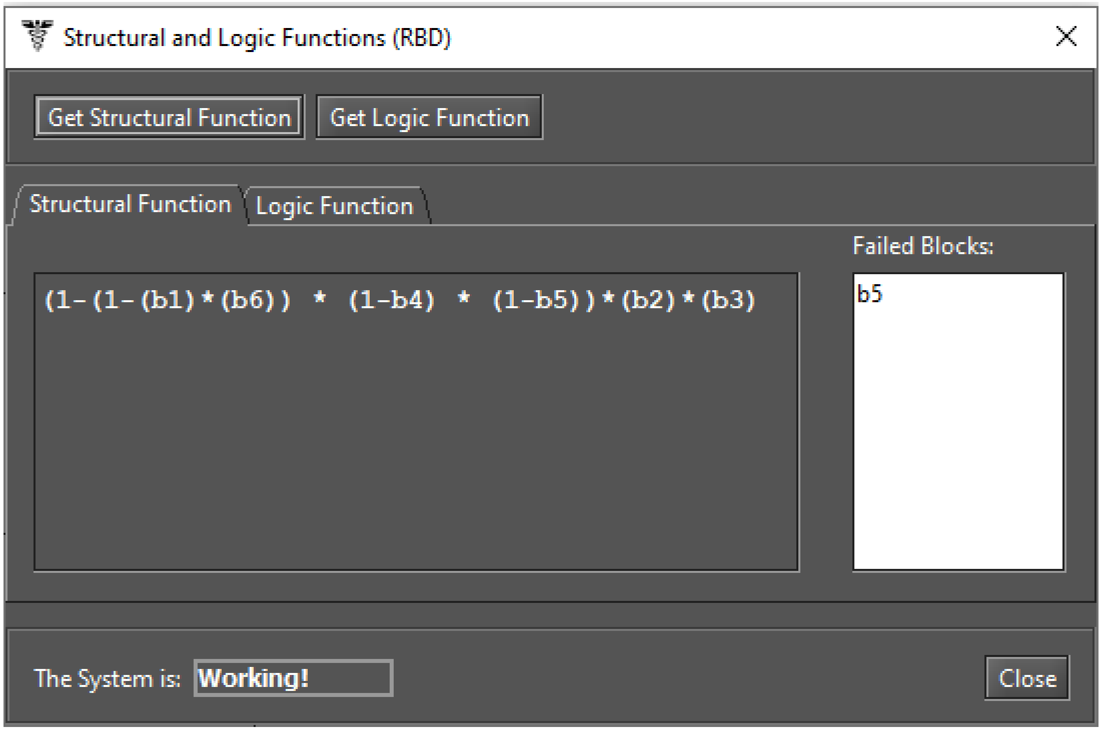


Figure 115: RBD Model

The structural and logic functions are accessible in the menu Evaluate -> RBD Evaluation -> Get Functions. Figure 116 and 117 show the structural and logic functions of the RBD model depicted above, respectively. In addition to the generated expressions, the tool indicates the blocks marked as failed (inoperative) and the current state of the system. As we can see, the failed block (b5) does not impact the system's operation.

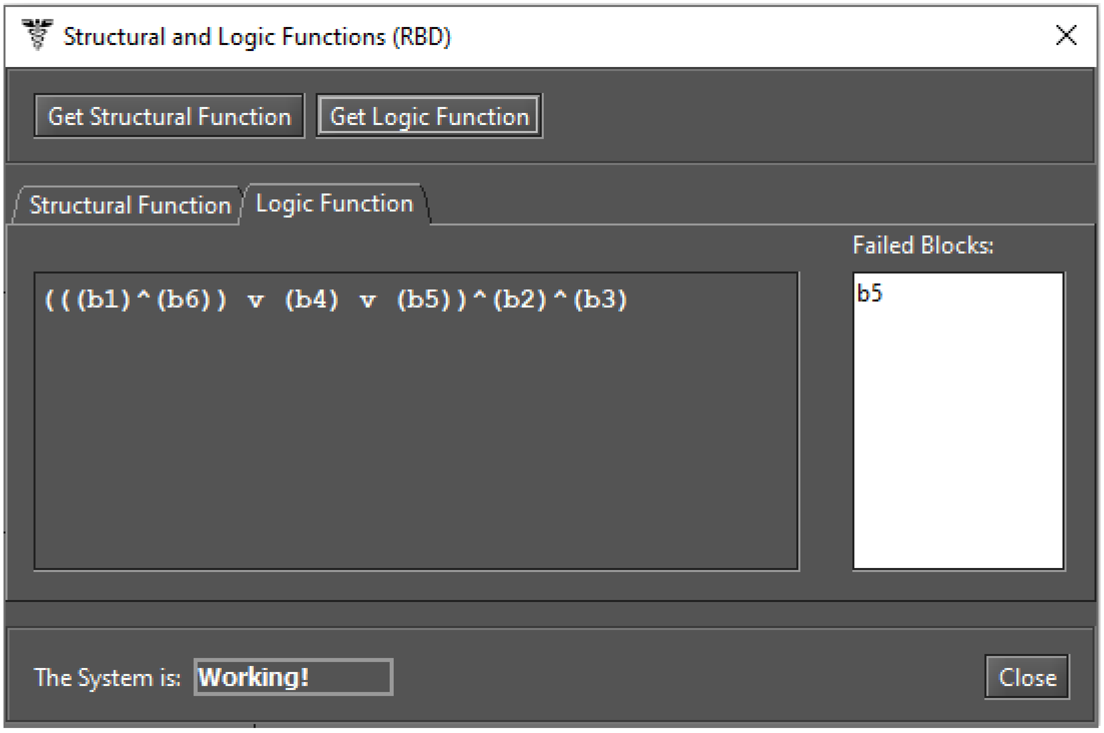


```
(1-(1-(b1)*(b6)) * (1-b4) * (1-b5))*(b2)*(b3)
```

Failed Blocks: b5

The System is: Working!

Figure 116: Structural Function



```
(((b1)^(b6)) v (b4) v (b5))^(b2)^(b3)
```

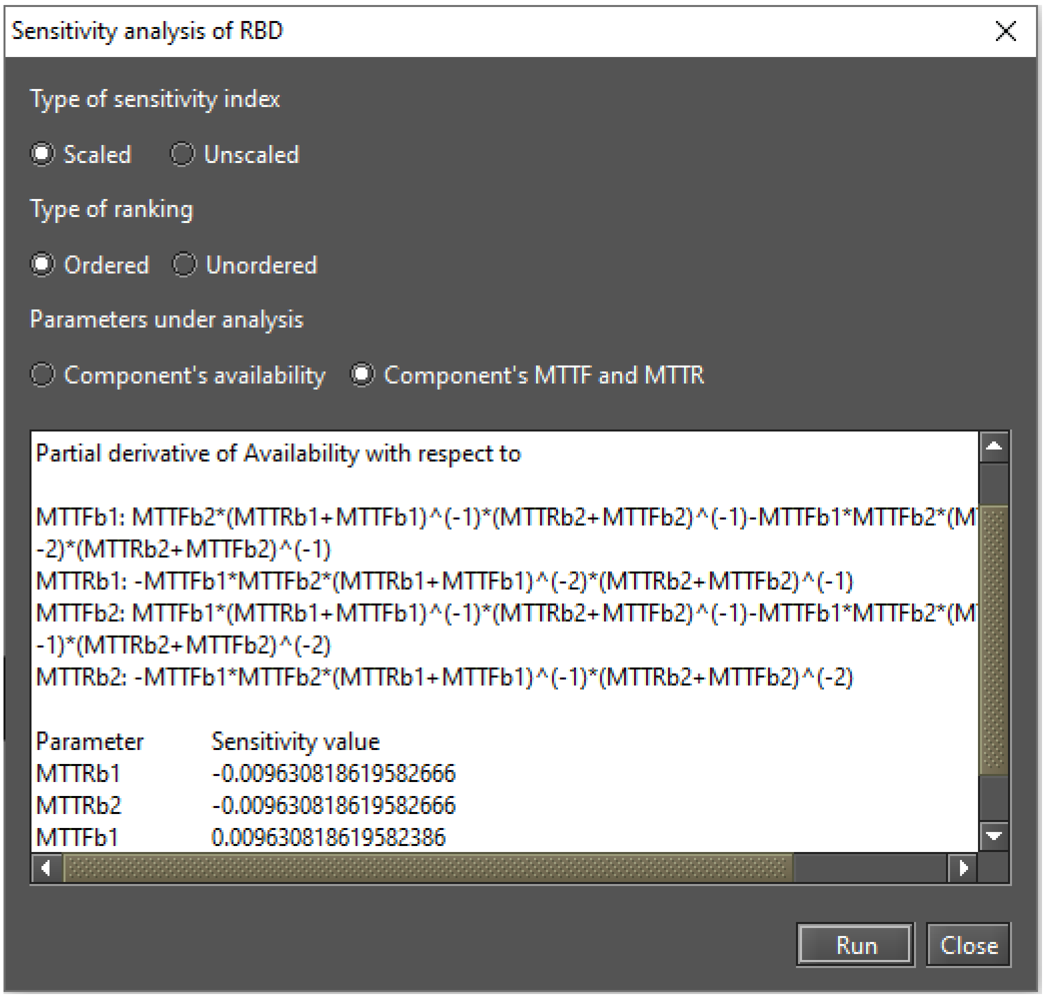
Failed Blocks: b5

The System is: Working!

Figure 117: Logic Function and the System State as Working

3.3.6 Sensitivity Analysis

The Mercury tool provides a functionality named Sensitivity Analysis that enables computing partial derivative sensitivity indices for RBD models. Those indices indicate the impact that every input parameter has on the model's availability. If the model has hierarchical blocks, the input parameters of the CTMC sub-model are also considered in the Sensitivity Analysis. This function is acessible in the menu Evaluate -> RBD Evaluation -> Sensitivity Analysis. Figure 118 shows the "RBD Sensitivity Analysis" window, which presents the partial derivative of the structural equation to every parameter, as well as the sensitivity indices. The sensitivity analysis is enabled only when the RBD model does not contain blocks with non-exponential failure or repair parameters.



Type of sensitivity index

[✓ Scaled] Unscaled

Type of ranking

[✓ Ordered] Unordered

Parameters under analysis

Component's availability [✓ Component's MTTF and MTTR]

```
Partial derivative of Availability with respect to

MTTFb1: MTTFb2*(MTTRb1+MTTFb1)^(-1)*(MTTRb2+MTTFb2)^(-1)-MTTFb1*MTTFb2*(M
-2)*(MTTRb2+MTTFb2)^(-1)
MTTRb1: -MTTFb1*MTTFb2*(MTTRb1+MTTFb1)^(-2)*(MTTRb2+MTTFb2)^(-1)
MTTFb2: MTTFb1*(MTTRb1+MTTFb1)^(-1)*(MTTRb2+MTTFb2)^(-1)-MTTFb1*MTTFb2*(M
-1)*(MTTRb2+MTTFb2)^(-2)
MTTRb2: -MTTFb1*MTTFb2*(MTTRb1+MTTFb1)^(-1)*(MTTRb2+MTTFb2)^(-2)

Parameter      Sensitivity value
MTTRb1         -0.009630818619582666
MTTRb2         -0.009630818619582666
MTTFb1         0.009630818619582386
```

Figure 118: Sensitivity Analysis of an RBD Model

4 CTMC Modeling and Evaluation

In the CTMC view, the user must click on the “Add new state” button, available on the CTMC toolbar (see Figure 119), and then she must click on the drawing area in order to create new CTMC states.

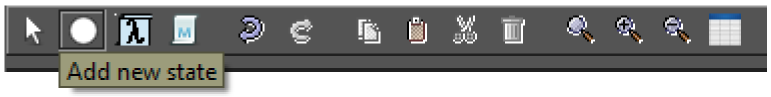


Figure 119: Adding a CTMC State

After adding the states, transitions between them are drawn by clicking in the center of the source state, when the mouse cursor changes to a hand, and dragging the line until the target state, as depicted in Figure 120. After that, a directed arc is created between the two states as depicted by Figure 121.

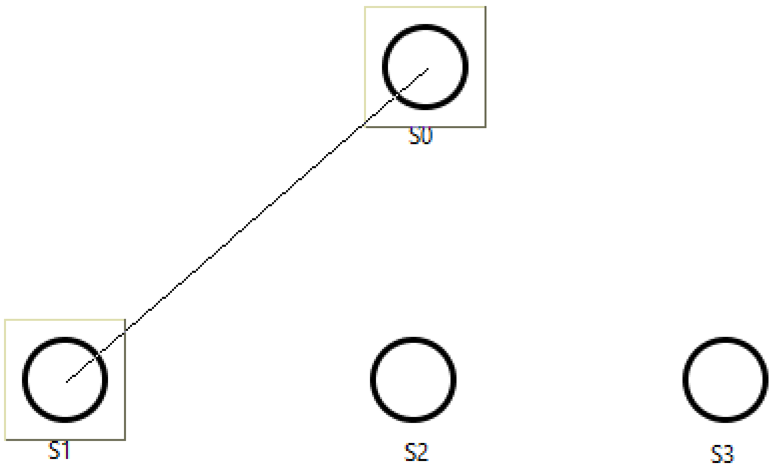


Figure 120: Adding a Transition to the CTMC

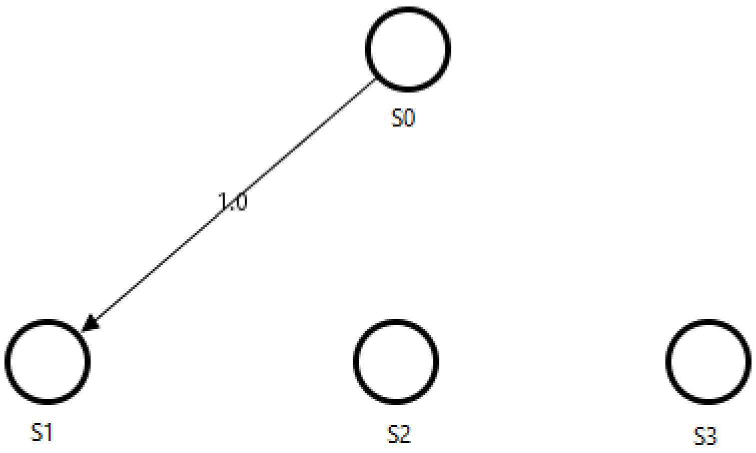
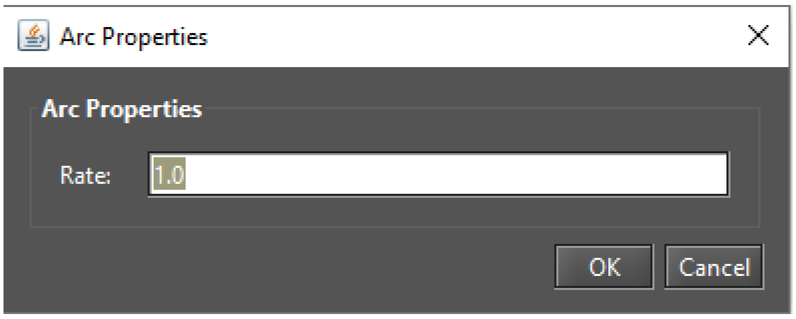


Figure 121: Transition between Two States

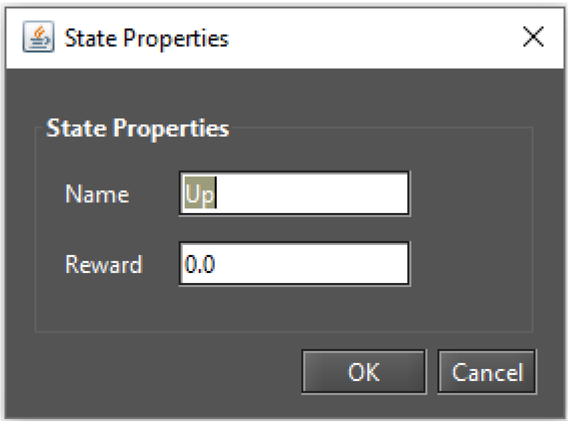
The user can define the rate of each transition by double-clicking in the respective arc or right-clicking on it and clicking in the "Properties" menu item in the pop-up menu that appears. Figure 122 shows the window where the transition rate can be defined.



Rate: 1.0

Figure 122: Defining the Rate for a CTMC State Transition

Mercury also allows assigning reward rates to the CTMC states. The user must double-click the selected state, or right-click on it and click on the "Properties" menu item on the pop-up menu that appears. Figure 123 shows the "State Properties" dialog box where the reward rate can be assigned to a CTMC state. The default reward for every state is zero. The user can enter any real value or expression containing user-defined parameters. The state name can also be changed in the same window.



Name: Up
Reward: 0.0

Figure 123: Modifying State Properties (Name and Reward)

After all states and transitions are properly defined (see Figure 124), it is possible to carry out stationary and transient analyses.

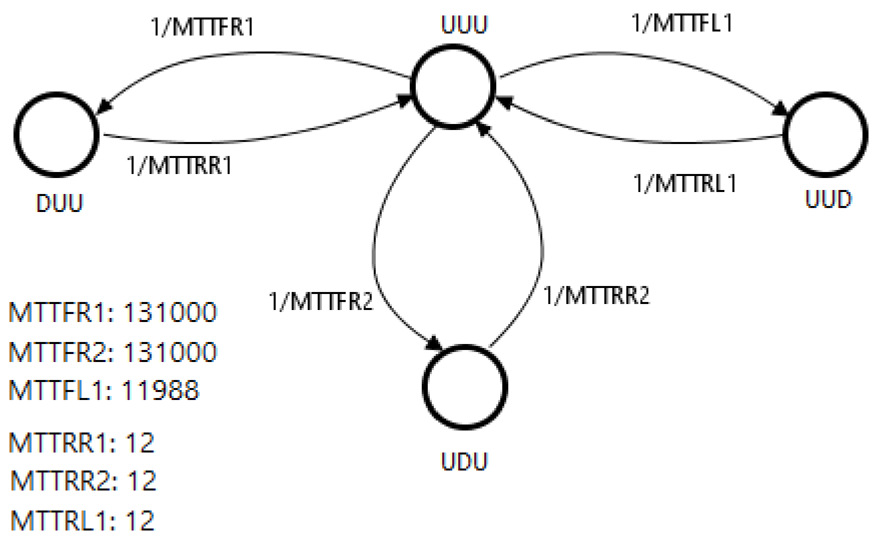


Figure 124: Example of CTMC Model

The user can export the infinitesimal generator matrix (i.e., the transition rate matrix) related to the model to a text file, by clicking in the matrix icon shown in Figure 125.

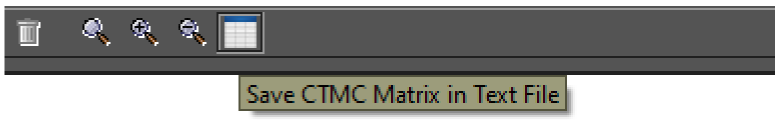


Figure 125: Exporting the CTMC Transition Rate Matrix

4.1 Input Parameters

The transition rates of CTMCs can be defined by using expressions containing both numbers and user-defined parameters. The “Add new parameter” button on the CTMC toolbar (see Figure 126) is represented by a λ icon, and it enables creating a symbolic parameter.

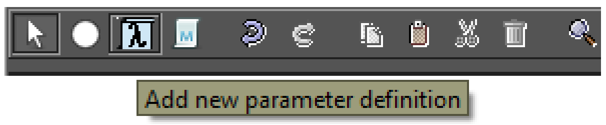
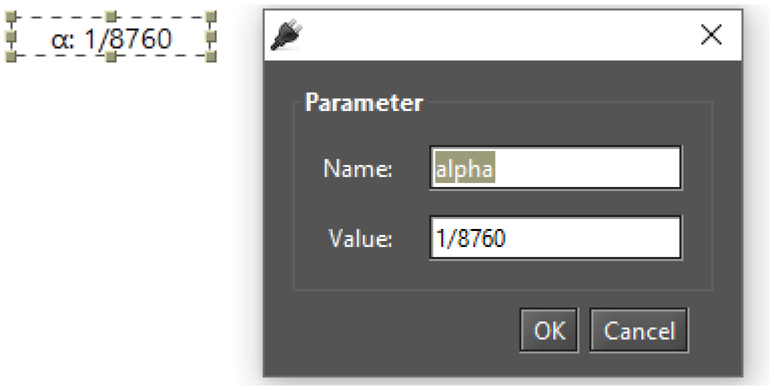


Figure 126: Adding a CTMC Parameter

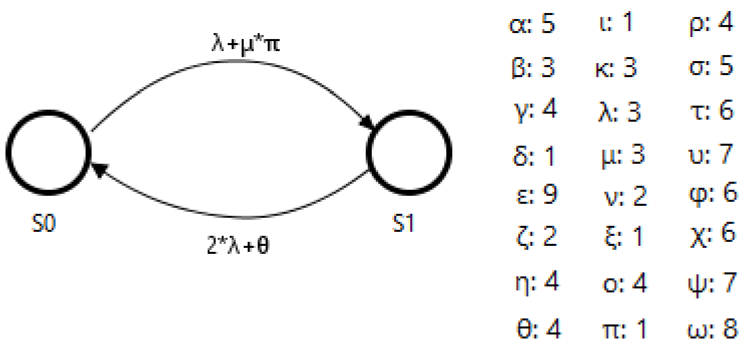
After clicking on that button, the user must click on any point in the CTMC drawing area. So, a new parameter is created with name **Param0** (or **Param1**, etc, if other parameters were already created). In order to change the parameter name and define a value for it, double-click it or go to the “Properties” menu item on the pop-up menu.

The name of the parameter can be defined by means of any combination of alphanumerical characters. Special characters (e.g., underscore, hyphen, ampersand) must be avoided. If names of Greek letters are used for parameters, Mercury converts them to the proper symbol on the Greek lowercase alphabet (see Figures 127 and 128). The value assigned to the parameter can be a numerical expression. Any symbols or other parameter names are not allowed in the parameter value.



Name: alpha
Value: 1/8760

Figure 127: Modifying a CTMC Parameter



α: 5	ι: 1	ρ: 4
β: 3	κ: 3	σ: 5
γ: 4	λ: 3	τ: 6
δ: 1	μ: 3	υ: 7
ε: 9	ν: 2	φ: 6
ζ: 2	ξ: 1	χ: 6
η: 4	ο: 4	ψ: 7
θ: 4	π: 1	ω: 8

Figure 128: Parameter Names defined with Greek Letters

4.2 Metrics

The “Add new metric” button (see Figure 129) enables defining metrics for the CTMC model.

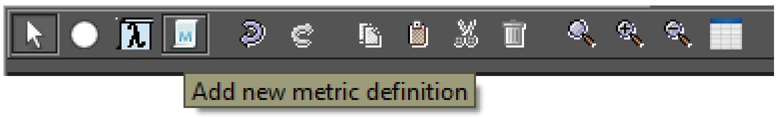


Figure 129: Adding a CTMC Metric

After inserting a metric, the user can change its name and define the expression that will be used to compute its value. The metric expression syntax is based on state probabilities (**P**{***state-name***}) and rewards (**R**{***state-name***}). The probabilities and rewards of any states can be combined (i.e., added, subtracted, and so on) in the expression. In the example depicted by Figure 130, the metric **AvB3** indicates the availability of the system (state “Up”) represented by the two-state CTMC.

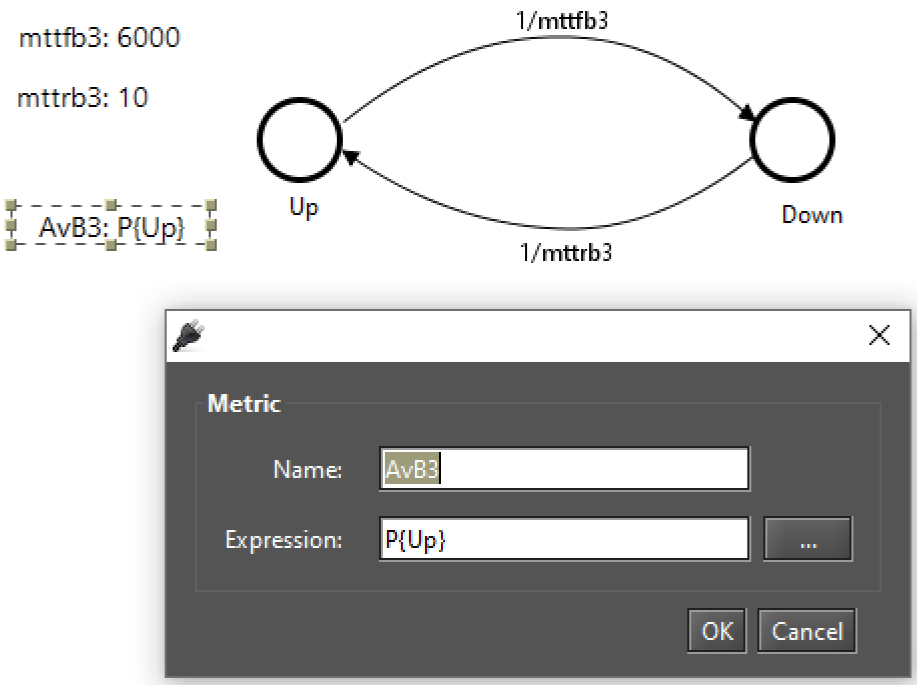


Figure 130: Defining the Metric Name and Expression

The availability will be computed from the expression **P{Up}**, that is the probability of being in state **Up**. When the model is evaluated through stationary or transient analysis, the metric value will be updated and displayed in the drawing area, as shown in Figure 131.

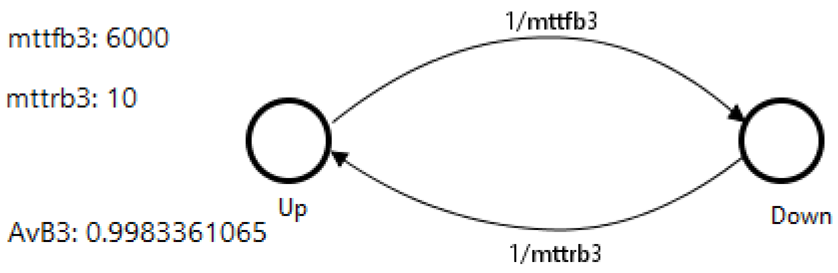
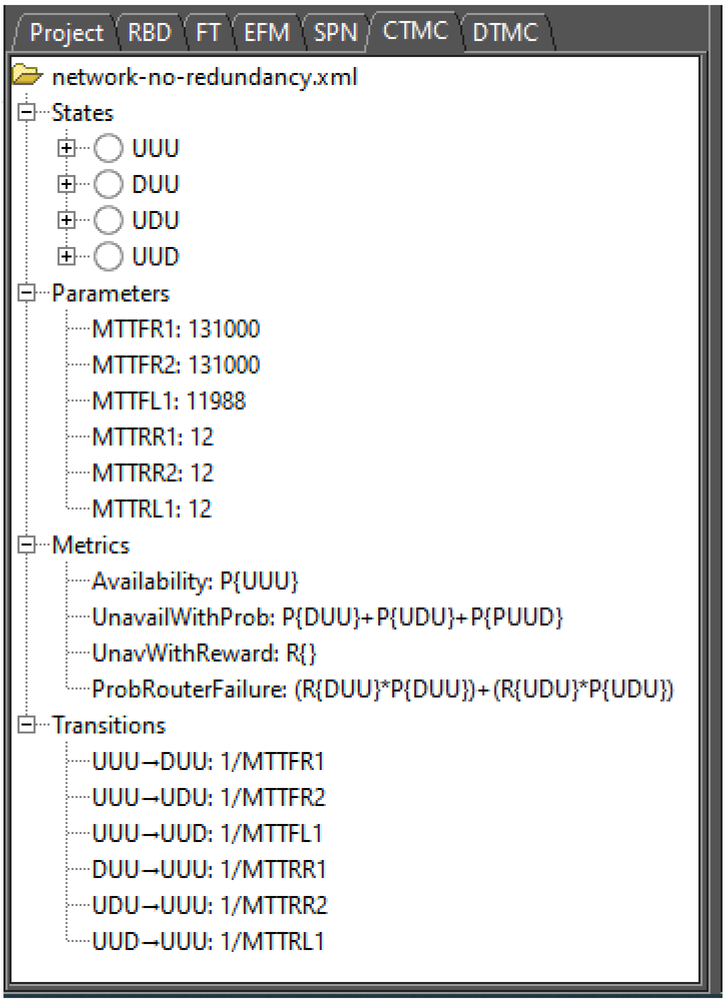


Figure 131: Metric Solved

The system reward rate can be obtained by using the metric expression **R{}**, and the mean time to absorption (when there is at least one absorbing state) can be computed by defining a metric having the expression **MTTA**.

The metric expression is still visible in the “Metrics” group, in the left-side panel on the CTMC tab (see Figure 132). That panel shows all components of the CTMC model: states, rewards, parameters, metrics, and state transitions. Also, every transition is represented in the left-side panel by the source and target states, followed by the expression or value assigned to that transition.



```
network-no-redundancy.xml
States
    UUU
    DUU
    UDU
    UUD
Parameters
    MTTFR1: 131000
    MTTFR2: 131000
    MTTFL1: 11988
    MTTRR1: 12
    MTTRR2: 12
    MTTRL1: 12
Metrics
    Availability: P{UUU}
    UnavailWithProb: P{DUU}+P{UDU}+P{PUUD}
    UnavWithReward: R{}
    ProbRouterFailure: (R{DUU}*P{DUU})+(R{UDU}*P{UDU})
Transitions
    UUU→DUU: 1/MTTFR1
    UUU→UDU: 1/MTTFR2
    UUU→UUD: 1/MTTFL1
    DUU→UUU: 1/MTTRR1
    UDU→UUU: 1/MTTRR2
    UUD→UUU: 1/MTTRL1
```

Figure 132: CTMC Left-Side Panel

4.3 CTMC Evaluation

The Mercury tool makes it possible to carry out a large number of evaluations on CTMCs. The tool provides three functionalities for CTMC evaluations: "Stationary Analysis", "Transient Analysis", and "Sensitivity Analysis". These evaluations are available in the menu Evaluate -> CTMC Evaluation, as we can see in Figure 133. In the next subsections, we will introduce each one of these evaluations.

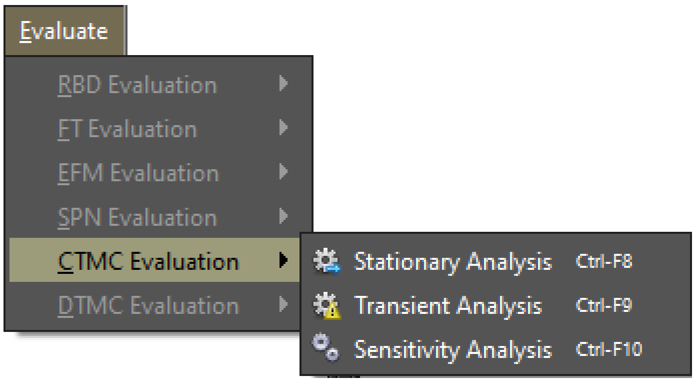


Figure 133: CTMC Evaluation Menu

4.3.1 CTMC Stationary Analysis

Stationary Analysis computes steady-state probabilities, useful for evaluating the long-term average behavior of modeled systems. Figure 134 shows the "Stationary Analysis" window, which has a combo box for selecting one of two solution methods available: **Direct - GTH** (Grassmann-Taksar-Heyman) or **Iterative - Gauss-Seidel**.

When solving the CTMC through GTH, the user can change the **maximum error** used in the algorithm. The default value for the maximum error is 0.0000001 (10^-7). By clicking on the "Run" button, the solution algorithm is triggered and as soon as it finishes, the results are presented in the text area at the bottom of the window (see Figure 135), and written in a plain text file with the name of the project file appended with the "-StationaryAnalysis.txt" suffix.

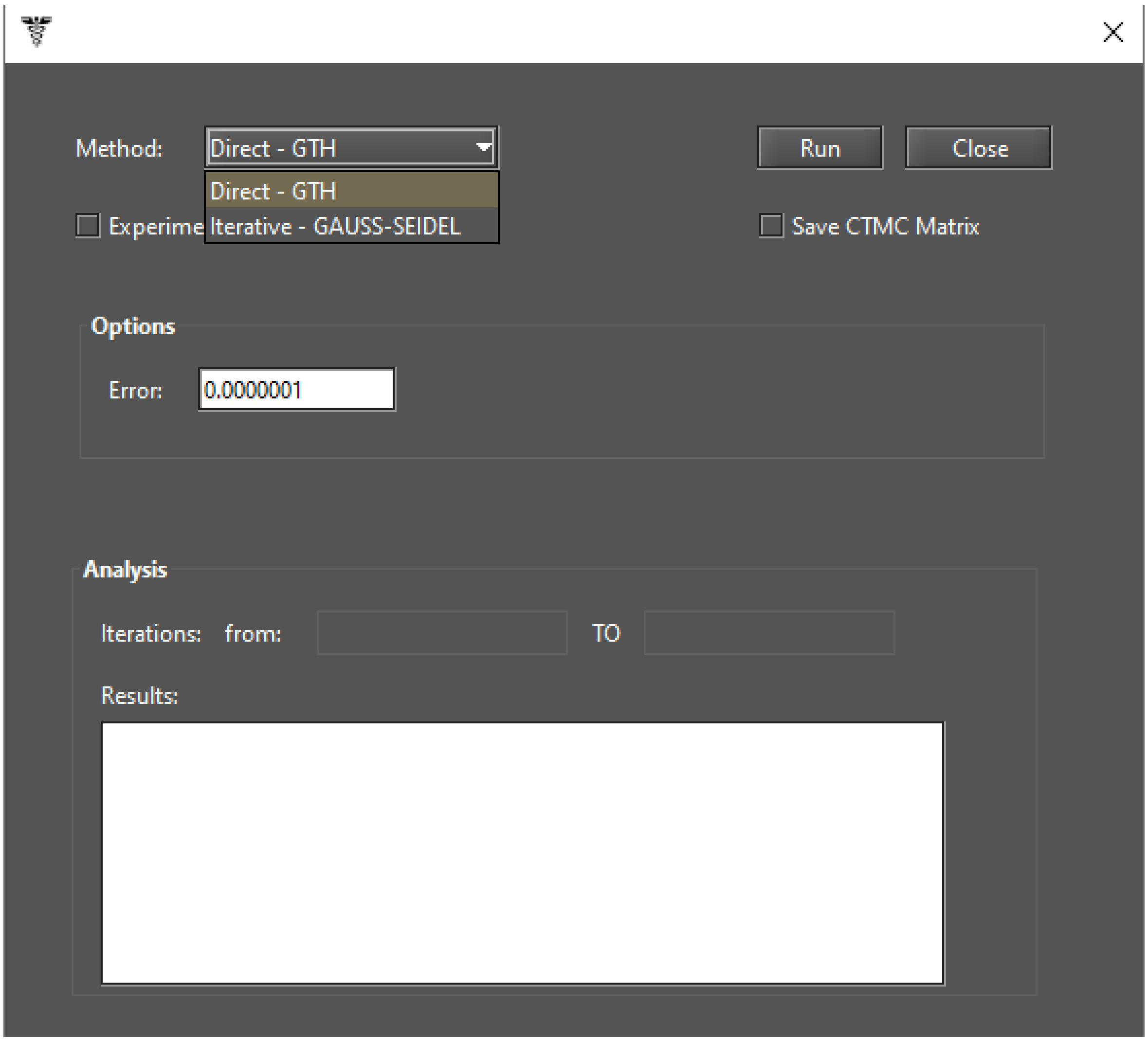
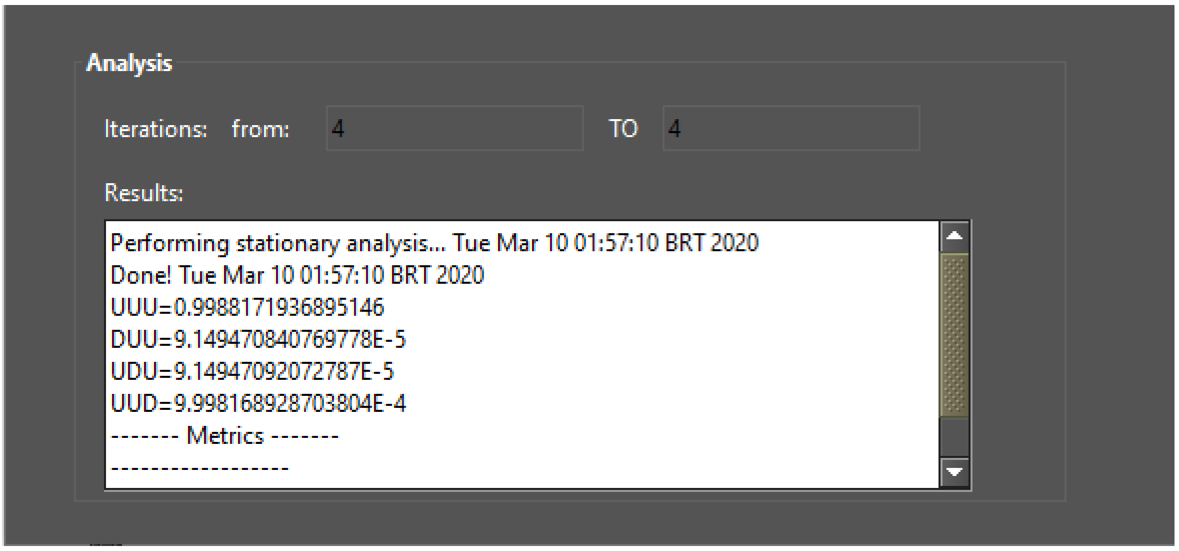


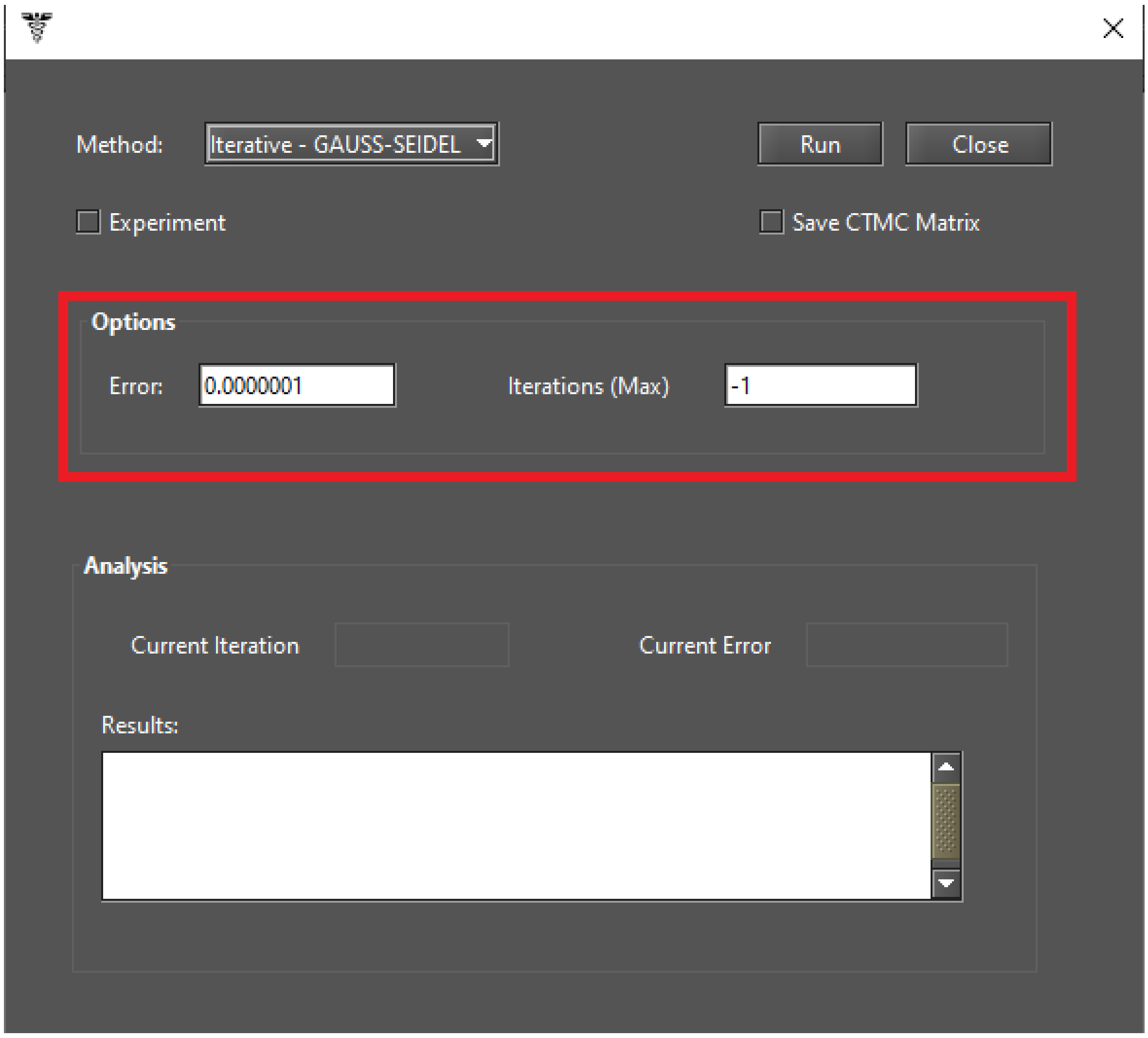
Figure 134: Stationary Analysis Window



```
Performing stationary analysis... Tue Mar 10 01:57:10 BRT 2020
Done! Tue Mar 10 01:57:10 BRT 2020
UUU=0.9988171936895146
DUU=9.149470840769778E-5
UDU=9.14947092072787E-5
UUD=9.998168928703804E-4
------- Metrics -------
-----------------
```

Figure 135: Stationary Analysis Results

When solving the CTMC through Gauss-Seidel, besides the maximum error, the user can also change the maximum number of iterations. The default value for such a parameter is “-1”, indicating that the algorithm only stops when the convergence of results is reached, considering the entered error (see Figure 136).


Method: Iterative - GAUSS-SEIDEL

Options

Error: 0.0000001

Iterations (Max) -1

Figure 136: Stationary Analysis Window - Gauss-Seidel Method

The metrics are updated in the CTMC drawing area as soon as the analysis is completed, independently of the chosen solution method, as depicted by Figure 137, where the metrics are on the right side of the model — Availability, UnavWithReward, UnavailWithProb, and ProbRouterFailure.

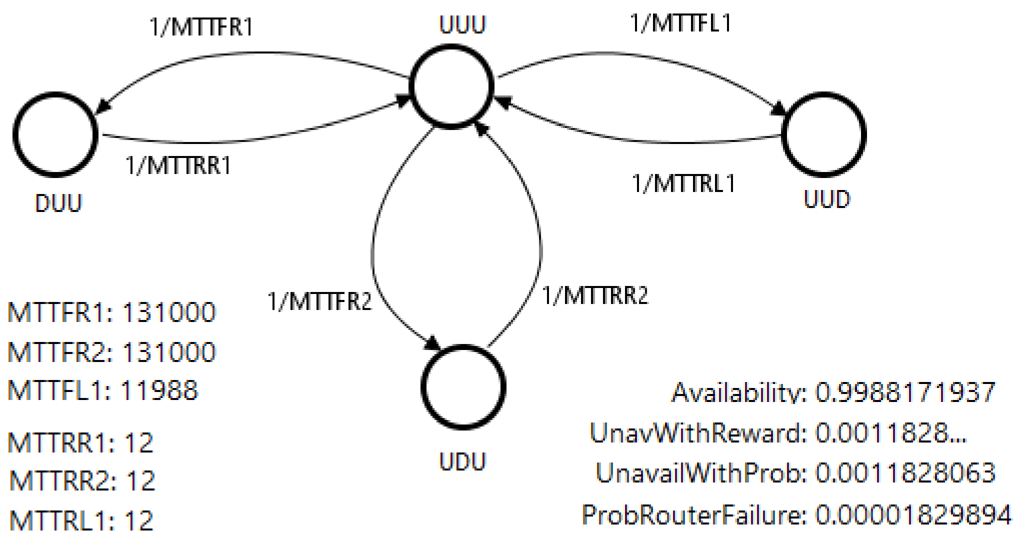
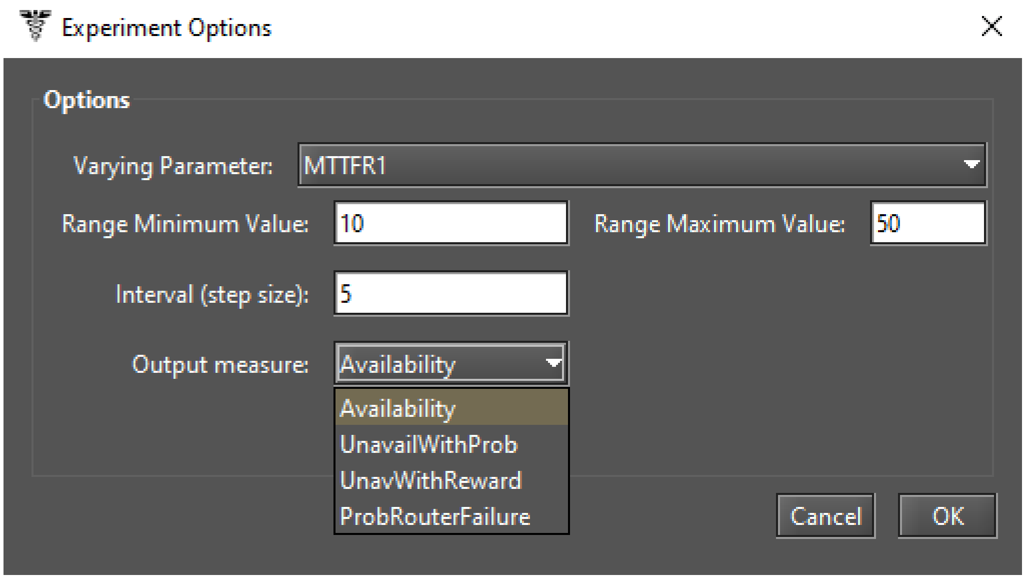


Figure 137: CTMC Metrics after the Model Solution

The CTMC model can also be solved for a range of values of user-defined parameters. This is accomplished by clicking in the “Experiment” checkbox on the “Stationary Analysis” window and then on the “Run” button. A new dialog is displayed in order to define the options for the experiment to be carried out (see Figure 138).



Experiment Options

Options

Varying Parameter: MTTFR1

Range Minimum Value: 10

Range Maximum Value: 50

Interval (step size): 5

Output measure: Availability

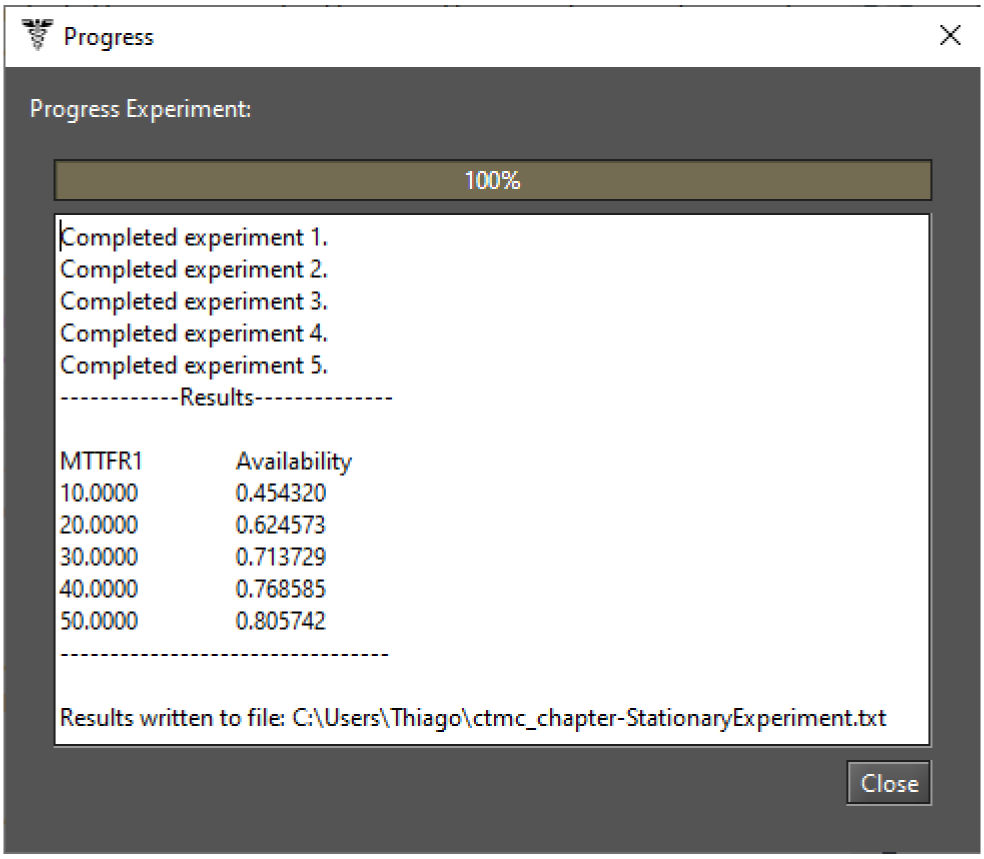
Figure 138: CTMC Experiment

Below, we describe each of them.

- **Varying Parameter.** The parameter to have its value changed.
- **Output measure.** The metric to be evaluated.
- **Range Minimum Value.** The initial value to be assigned to the selected parameter.
- **Range Maximum Value.** The final value to be assigned to the selected parameter.

- **Interval.** It is the step-size for changing the value of the parameter. The parameter starts with the minimum value and its value is increased by considering the entered interval. At each change, the selected metric is evaluated. The experiment ends when the maximum value for the parameter is reached.

At the end of the experiment, the results are shown and a graph is plotted, as we can see by looking at Figures 139 and 140, respectively.



```
Completed experiment 1.
Completed experiment 2.
Completed experiment 3.
Completed experiment 4.
Completed experiment 5.
------------Results-------------

MTTFR1          Availability
10.0000         0.454320
20.0000         0.624573
30.0000         0.713729
40.0000         0.768585
50.0000         0.805742
---------------------------------

Results written to file: C:\Users\Thiago\ctmc_chapter-StationaryExperiment.txt
```

Figure 139: CTMC Experiment Results

Another checkbox in the “Stationary Analysis” dialog allows saving the CTMC matrix to a file just after the model solution. It is written in a plain text file with the name of the project file appended with the “-MatrixQ.txt” suffix.

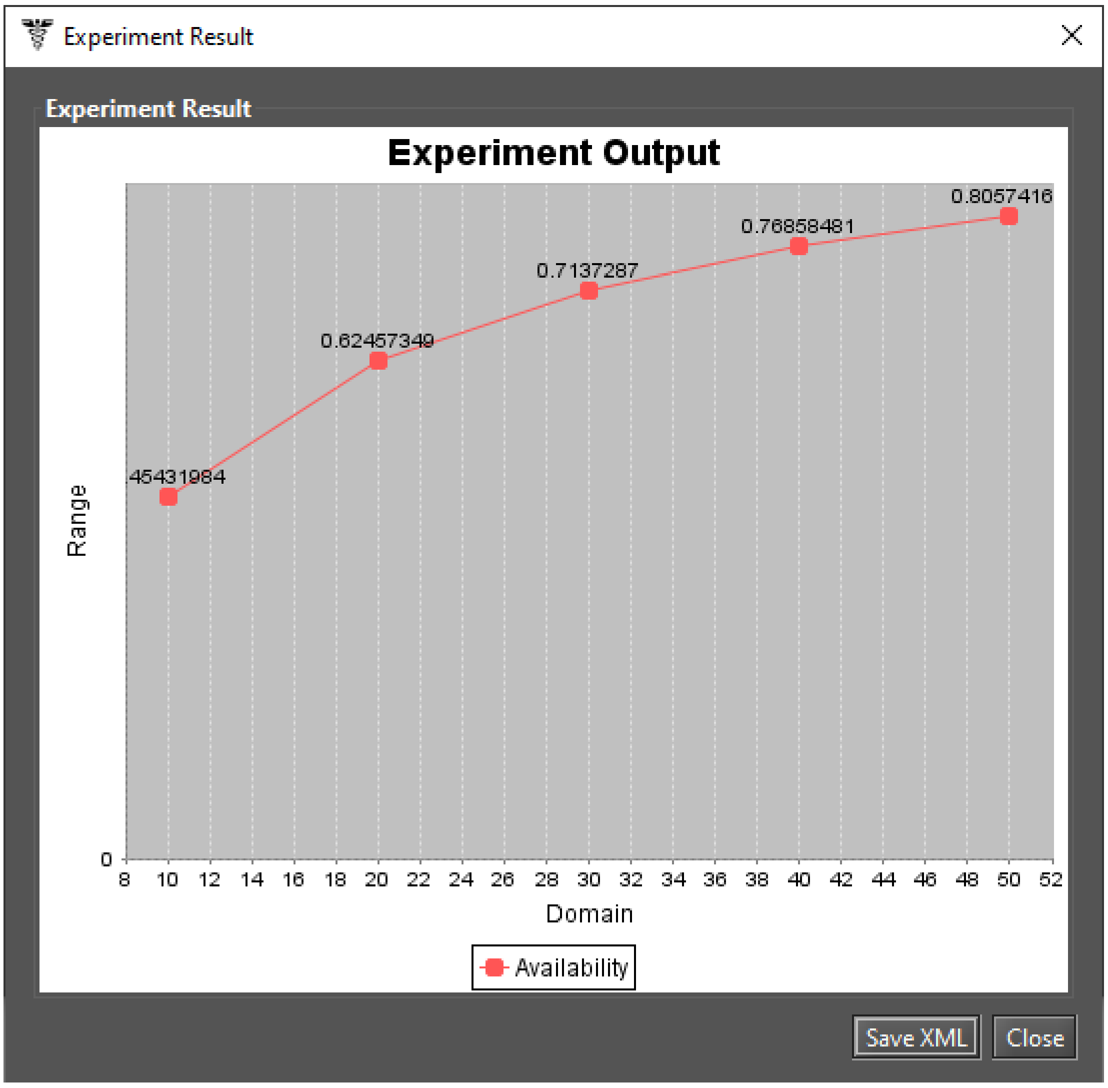


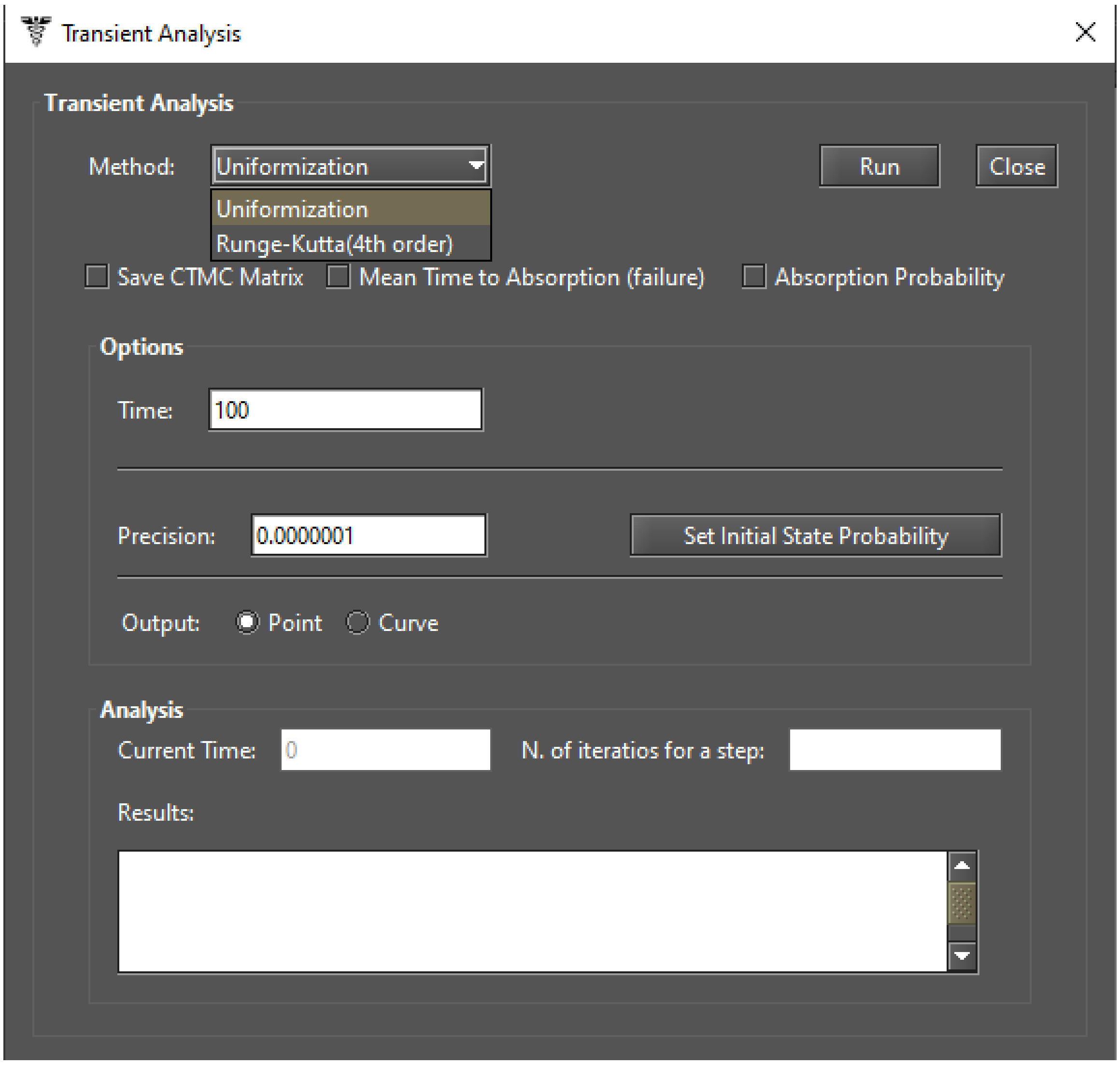
Figure 140: CTMC Experiment Graph

4.3.2 CTMC Transient Analysis

Transient Analysis computes time-dependent probabilities, useful for evaluating the behavior of modeled systems at a particular point in time. Figure 141 shows the "Transient Analysis" window, which has a combo box for selecting one of the two solution methods available: **Uniformization** (also known as Jensen's method) and **Runge-Kutta (4th order)**.

When solving the CTMC, the user can define:

- The **time** for which the analysis will be carried out (default: 100).
- The **precision** of results (default:10^-7).
- The **initial state probabilities** (default: 1 for the first inserted state, 0 for the remaining states). These probabilities are defined by clicking on the "Set Initial State Probability" button (see Figure 142).


Transient Analysis

Transient Analysis

Method: Uniformization

Uniformization
Runge-Kutta(4th order)

Run | Close

☐ Save CTMC Matrix ☐ Mean Time to Absorption (failure) ☐ Absorption Probability

Options

Time: 100

Precision: 0.0000001

Set Initial State Probability

Output: [✓ Point] Curve

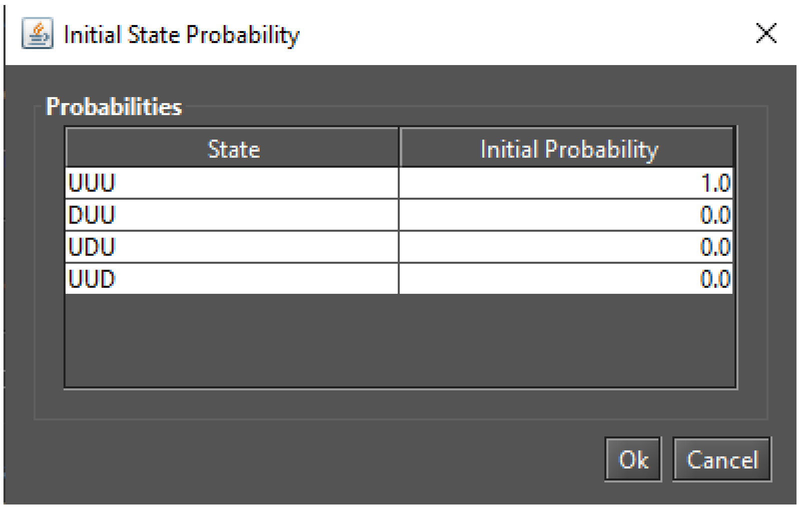
Analysis

Current Time: 0

N. of iteratios for a step:

Results:

Figure 141: Transient Analysis Window



State	Initial Probability
UUU	1.0
DUU	0.0
UDU	0.0
UUD	0.0

Figure 142: Initial State Probability Dialog

When selecting the Uniformization method, note that the time required for obtaining the results is proportional to the time entered for the analysis because Uniformization is an iterative algorithm.

By clicking on the “Run” button, the solution algorithm is triggered. As soon as it finishes, the results are presented in the text area at the bottom of the “Transient Analysis” window, also they are written in a plain text file having the filename of the project appended with the “-TransientAnalysis.txt” suffix.

The “Transient Analysis” window also allows the user to choose between a **Point** or **Curve** analysis. **Point** analysis is the default, and it shows the state probabilities only for the specific point in time. **Curve** analysis writes in a plain text file all state probabilities computed from time equals zero until the specified time.

The mean time to absorption (**MTTA**) is a metric that can be computed in the “Transient Analysis” window, by checking the “Mean Time to Absorption (failure)” checkbox. The MTTA value is presented after the state probabilities in the “Results” text area. For MTTA calculation, the user might also define a metric on the drawing area by using the keyword **MTTA** as its expression. The “Absorption Probability” for each state is another metric that is available in the transient evaluation.

4.3.3 Sensitivity Analysis

Sensitivity Analysis enables computing partial derivative sensitivity indices for CTMCs. Those indices indicate the impact that every input parameter has on a model's metric. This function is acessible in the menu Evaluate -> CTMC Evaluation -> Sensitivity Analysis. Figure 143 depicts a CTMC for representing the availability of a network comprising two routers and one link.

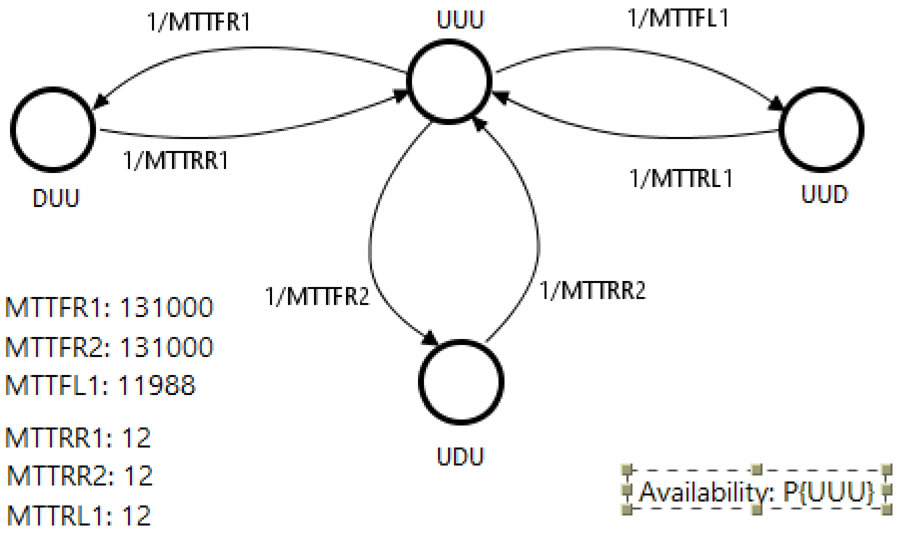


Figure 143: CTMC Model for a Network comprising Two Routers and One Link

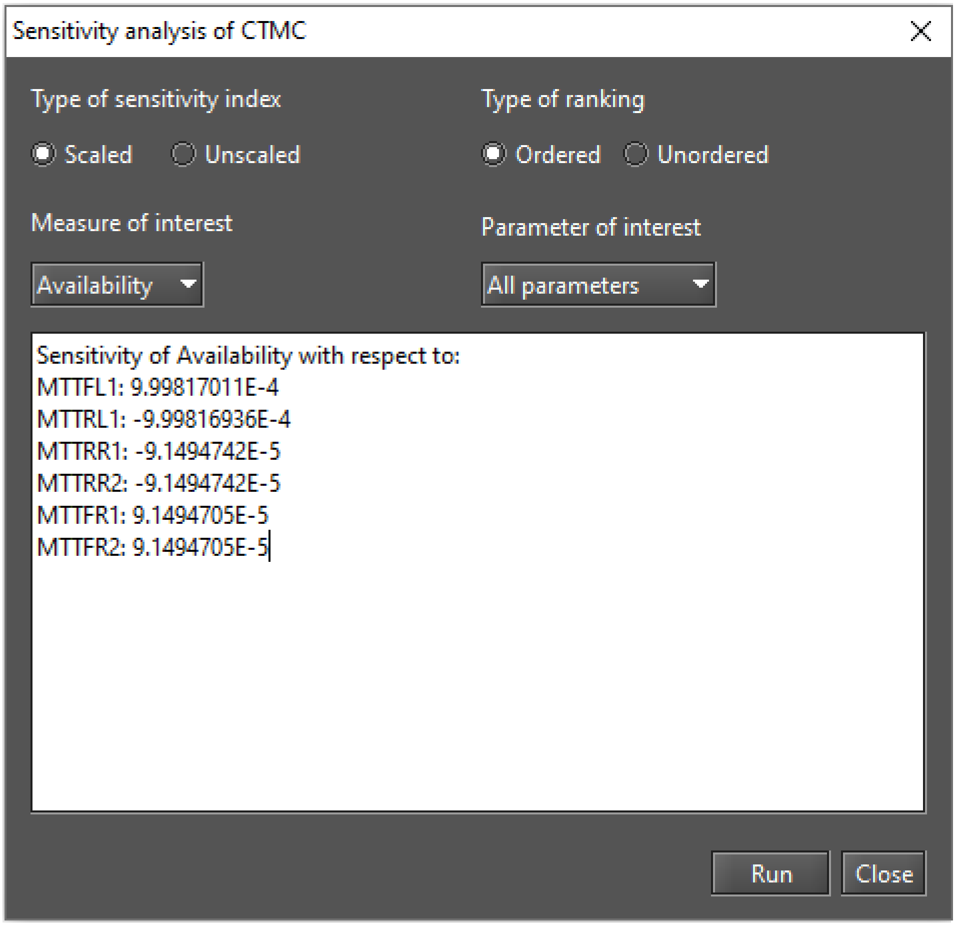
This model was proposed in [5], and it has six parameters that affect system availability. Those parameters are the mean time to failure (MTTF) and mean time to repair (MTTR) of each component: router 1 (R1), router 2 (R2), and link (L1). Their sensitivity ranking was computed by using Mercury as illustrated in Figure 144.

The "Sensitivity Analysis" window has four options:

- The **type of sensitivity index** can be scaled or unscaled. When the user chooses scaled indices, every partial derivative is multiplied by the ratio of the respective parameter value and metric value. This removes the influence of parameter units and provides sensitivity in a non-dimensional view. Unscaled indices are the raw results of partial derivatives. For more details on scaled and unscaled indices, please refer to [5] and [6].

- The **type of ranking** might be ordered or unordered. Usually ordered rankings are preferred for fast identification of most important parameters as well as those which have little impact on the chosen metric.

- The **measure of interest** can be any user-defined CTMC measure, for which the user is interested in assessing sensitivity to input parameters. Please notice that if no measure has been defined, no sensitivity analysis can be carried out. All measures can be evaluated at once.



Type of sensitivity index: [✓ Scaled] Unscaled

Type of ranking: [✓ Ordered] Unordered

Measure of interest: Availability

Parameter of interest: All parameters

```
Sensitivity of Availability with respect to:
MTTFL1: 9.99817011E-4
MTTRL1: -9.99816936E-4
MTTRR1: -9.1494742E-5
MTTRR2: -9.1494742E-5
MTTFR1: 9.1494705E-5
MTTFR2: 9.1494705E-5
```

Figure 144: Results of Sensitivity Analysis for a CTMC

- The **parameter of interest** can be any parameter in the model. The user must choose between viewing the sensitivity of the chosen measure with respect to just one parameter, or to all parameters.

5 DTMC Modeling and Evaluation

In the DTMC view, the user must click on the "Add new state" button, available on the DTMC toolbar (see Figure 145), and then she must click on the drawing area in order to create new DTMC states.

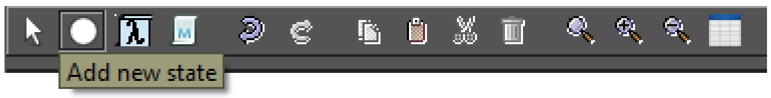


Figure 145: Adding a DTMC State

After adding the states, transitions between them are drawn by clicking in the center of the source state, when the mouse cursor changes to a hand, and dragging the line until the target state, as depicted by Figure 146. After that, a directed arc is created between the two states, as depicted in Figure 147.

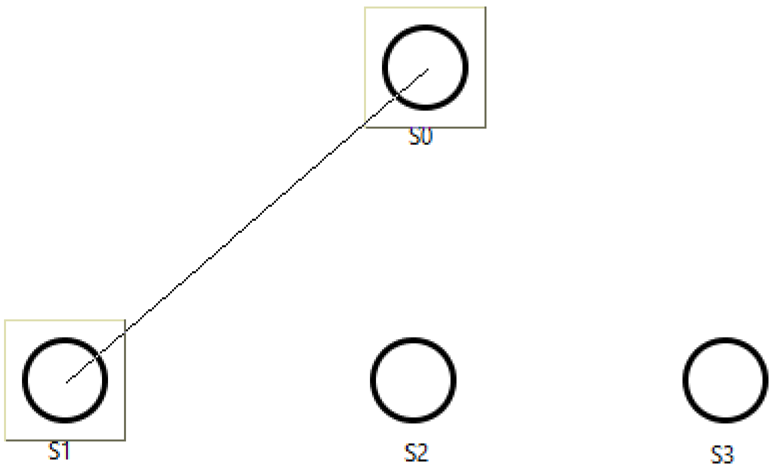


Figure 146: Adding a Transition Between States

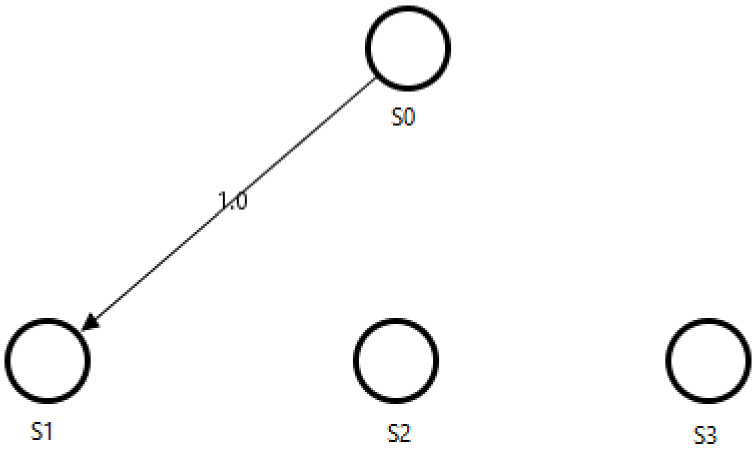


Figure 147: Transition between Two States

In DTMC, it is possible to define the probability of remaining in the current state once entering that state. In the Mercury tool, the user can define this by using self-loops. To add a self-loop to a state it is necessary to right-click on the DTMC state and click in the "Self-Loop" menu item in the pop-up menu that appears, as depicted by Figure 148.

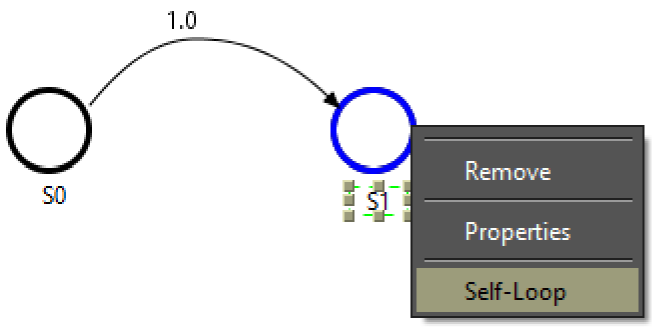


Figure 148: Adding a Self-Loop to a DTMC State

After that, a self-loop arc is drawn for the state in the drawing area, as we can see by looking at Figure 149, and the left-side panel on the DTMC tab is updated with the new transition.

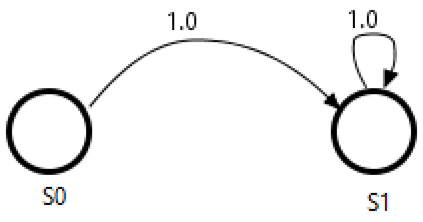
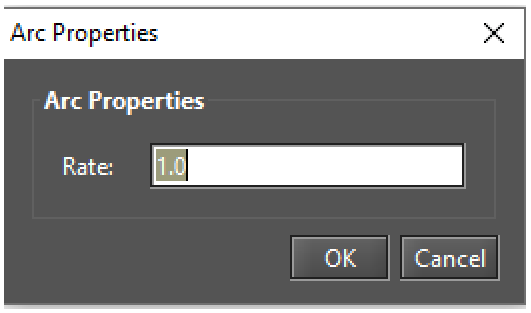


Figure 149: DTMC with a Self-Loop Transition

The user can define the probability of each transition by double-clicking in the respective arc or right-clicking on it and clicking in the "Properties" menu item in the pop-up menu that appears. Figure 150 shows the window where the transition probability can be defined.



Rate: 1.0

Figure 150: Defining the Probability for a DTMC State Transition

Figure 151 shows a DTMC model as an example having parameters and a metric defined. After all states and transitions are properly defined, it is possible to perform stationary and transient analyses.

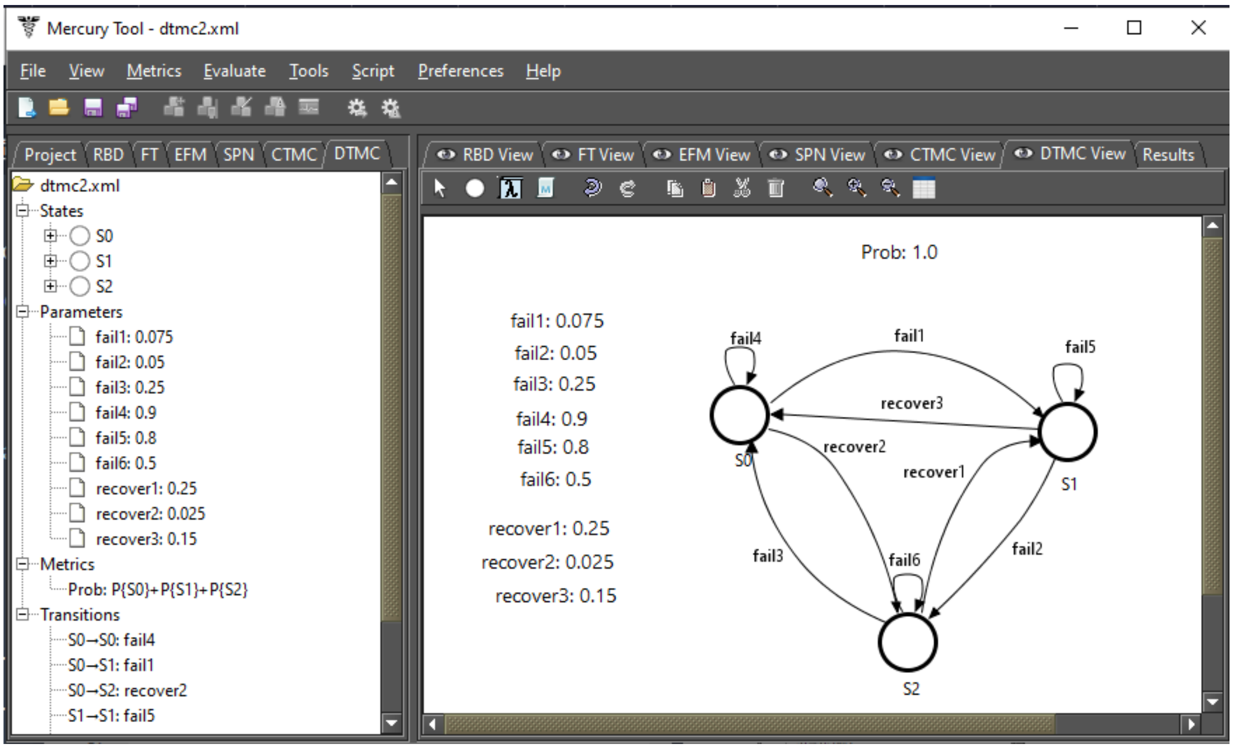


Figure 151: Example of a DTMC Model

The user can export the probability matrix related to the model to a text file, by clicking in the matrix icon shown in Figure 152.

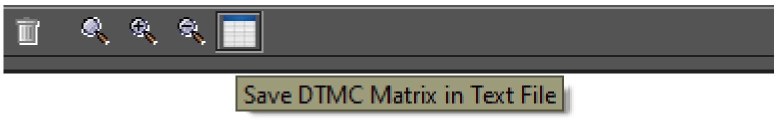
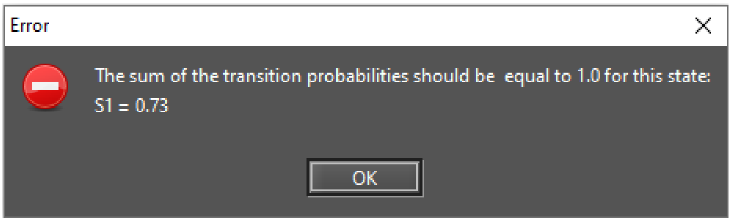


Figure 152: Exporting the DTMC Probability Matrix

It is worth mentioning that when modeling DTMCs, the sum of the probabilities of all output arcs of each state must be equal to one. When this does not occurs, it is not possible to carry out evaluations on the DTMC model by using the Mercury tool. Figure 153 shows the error message displayed when this condition is not met.


Error

The sum of the transition probabilities should be equal to 1.0 for this state:
S1 = 0.73

Figure 153: Transition Probabilities Error

5.1 Input Parameters

The transition probabilities of DTMCs can be defined by using expressions containing both numbers and user-defined parameters. The “Add new parameter” button on the DTMC toolbar (see Figure 154) is represented by a λ icon, and it enables creating a symbolic parameter.

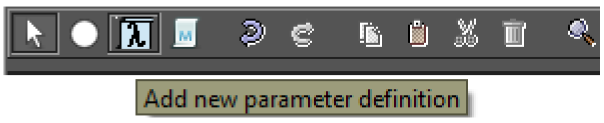
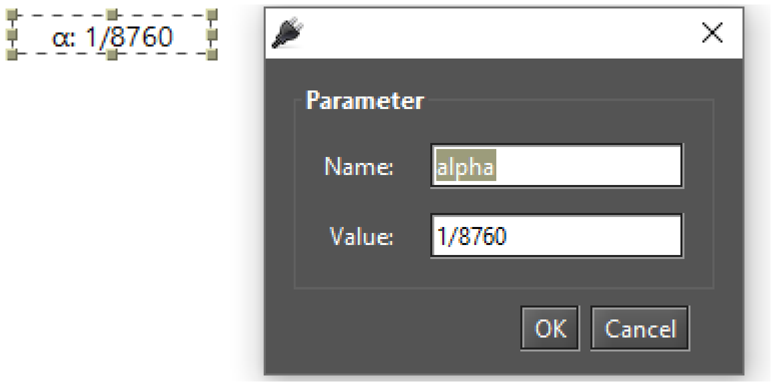


Figure 154: Adding a DTMC Parameter

After clicking on that button, the user must click on any point in the DTMC drawing area. So, a new parameter is created with name **Param0** (or **Param1**, etc, if other parameters were already created). In order to change the parameter name and define a value for it, double-click it or go to the “Properties” menu item on the pop-up menu.

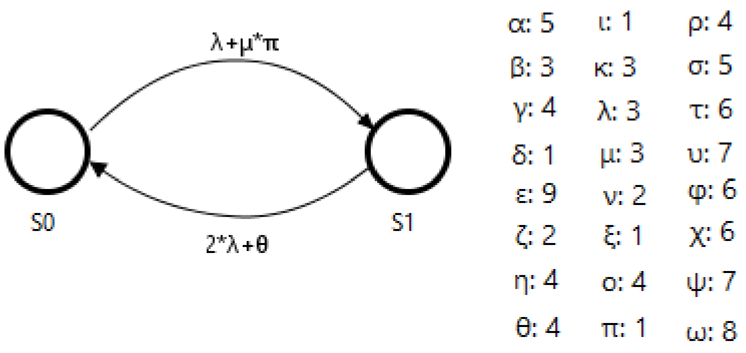
The name of the parameter can be defined by means of any combination of alphanumerical characters. Special characters (e.g., underscore, hyphen, ampersand) must be avoided. If names of Greek letters are used for parameters, Mercury converts them to the proper symbol on Greek lowercase alphabet (see Figures 155 and 156). The value assigned to the parameter can be a numerical expression. Any symbols or other parameter names are not allowed in the parameter value.



Name: alpha

Value: 1/8760

Figure 155: Modifying a DTMC Parameter



α: 5	ι: 1	ρ: 4
β: 3	κ: 3	σ: 5
γ: 4	λ: 3	τ: 6
δ: 1	μ: 3	υ: 7
ε: 9	ν: 2	φ: 6
ζ: 2	ξ: 1	χ: 6
η: 4	ο: 4	ψ: 7
θ: 4	π: 1	ω: 8

Figure 156: Parameter Names defined with Greek Letters

5.2 Metrics

The “Add new metric” button (see Figure 157) enables defining metrics for the DTMC model.

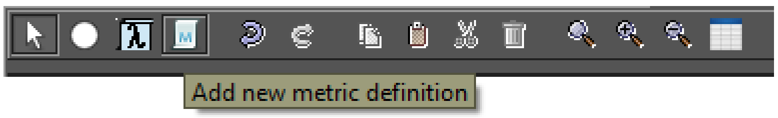
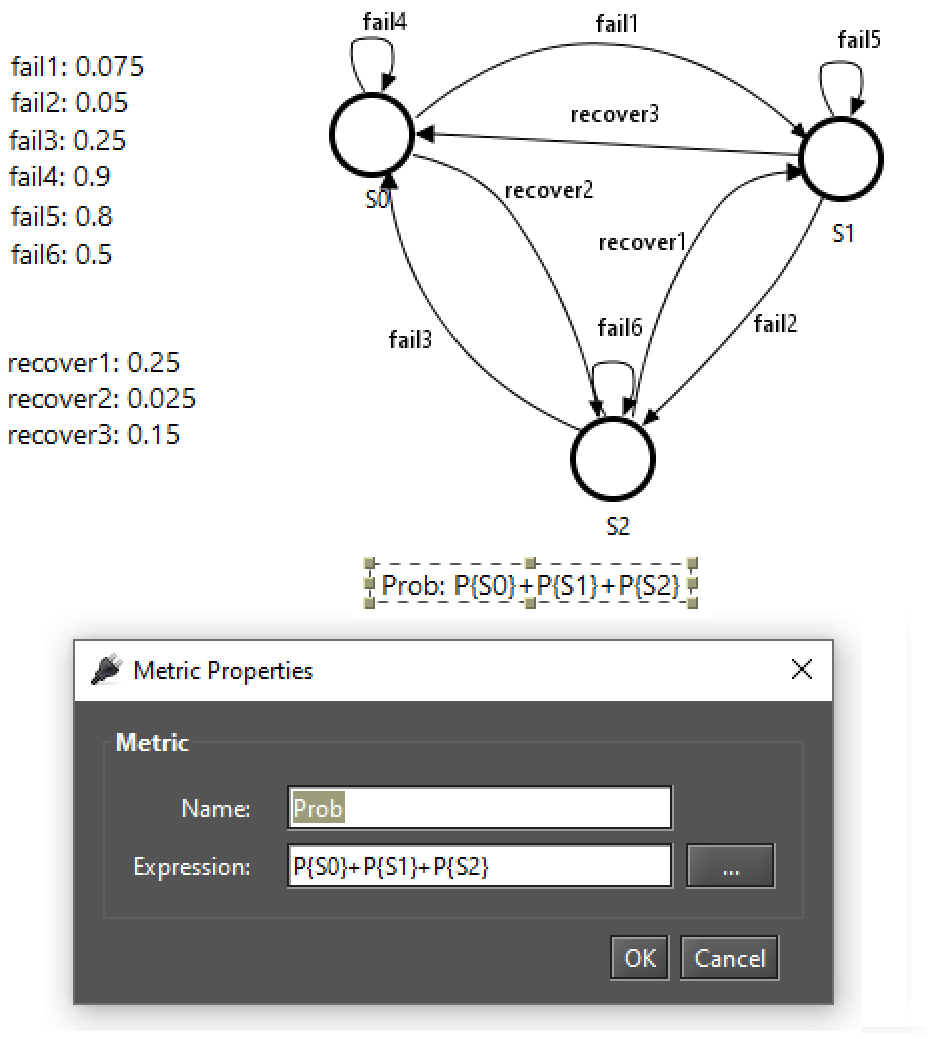


Figure 157: Adding a DTMC Metric

After inserting a metric, the user can change its name and define the expression that will be used to compute its value. The metric expression syntax is based on state probabilities (**P{*state-name*}**). The probabilities of any states can be combined (i.e., added, subtracted, and so on) in the expression. In the example depicted by Figure 158, the metric “Prob” is computed by considering the probability of the system remaining in states “S0”, “S1”, and “S2”.



Name: Prob

Expression: P{S0}+P{S1}+P{S2}

Figure 158: Defining the Metric Name and Expression

This metric value will be computed from the expression "**P{SO}+P{S1}+P{S2}**", that is the probability of being in the states "SO", "S1", and "S2". When the model is evaluated through stationary or transient analysis, the metric value will be updated and displayed in the drawing area, as shown by Figure 159.

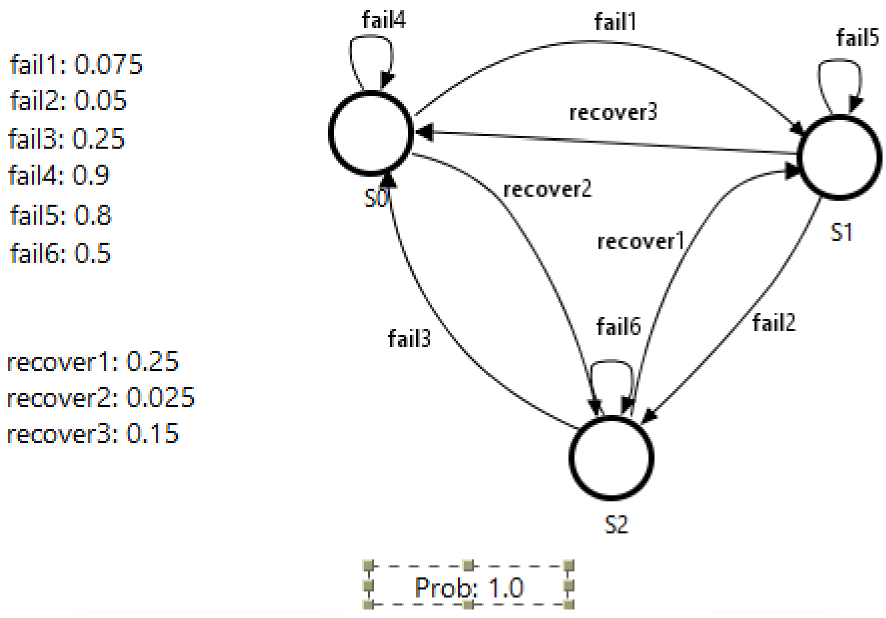


Figure 159: Metric Solved

The metric expression is still visible in the "Metrics" group, in the left-side panel on the DTMC tab (see Figure 160). That panel shows all components of the DTMC model: states, parameters, metrics, and state transitions. Also, every transition is represented in the left-side panel by the source and target states, followed by the expression or value assigned to that transition.

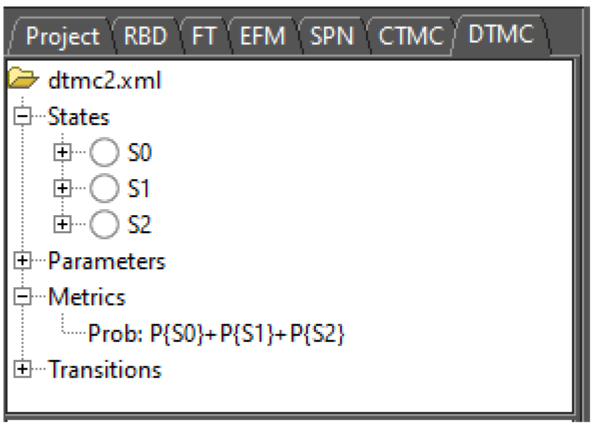


Figure 160: DTMC Left-Side Panel

5.3 DTMC Evaluation

The Mercury tool makes it possible to carry out a large number of evaluations on DTMCs. The tool provides two functionalities for DTMC evaluations: “Stationary Analysis” and “Transient Analysis”. These evaluations are available in the menu Evaluate -> DTMC Evaluation, as we can see in Figure 161. In the next subsections, we will introduce each one of these evaluations.

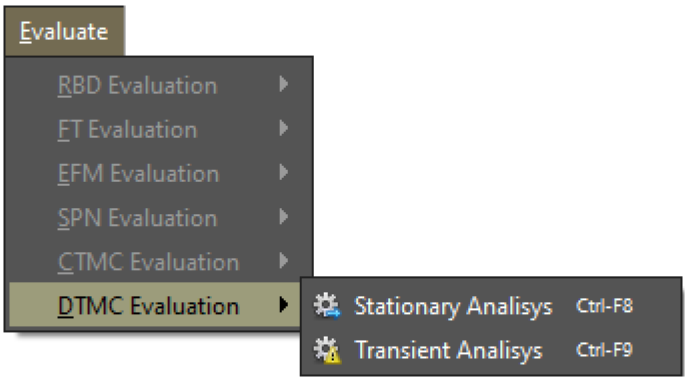
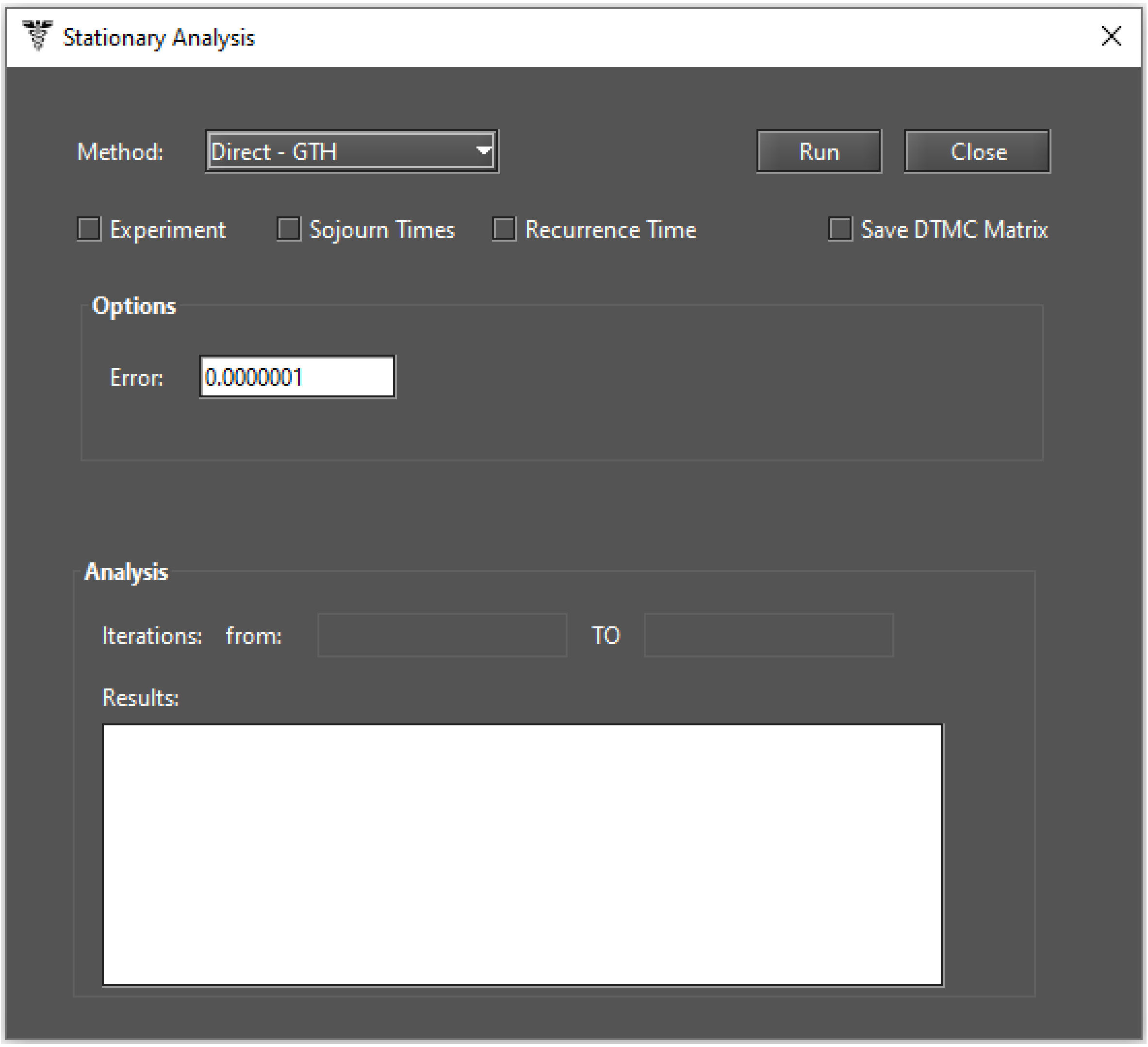


Figure 161: DTMC Evaluation Menu

5.3.1 DTMC Stationary Analysis

Stationary Analysis computes steady-state probabilities, useful for evaluating the long-term average behavior of modeled systems. Figure 162 shows the “Stationary Analysis” window, which has a combo box for selecting one of two solution methods available: **Direct - GTH** (Grassmann-Taksar-Heyman) or **Iterative - Gauss-Seidel**.

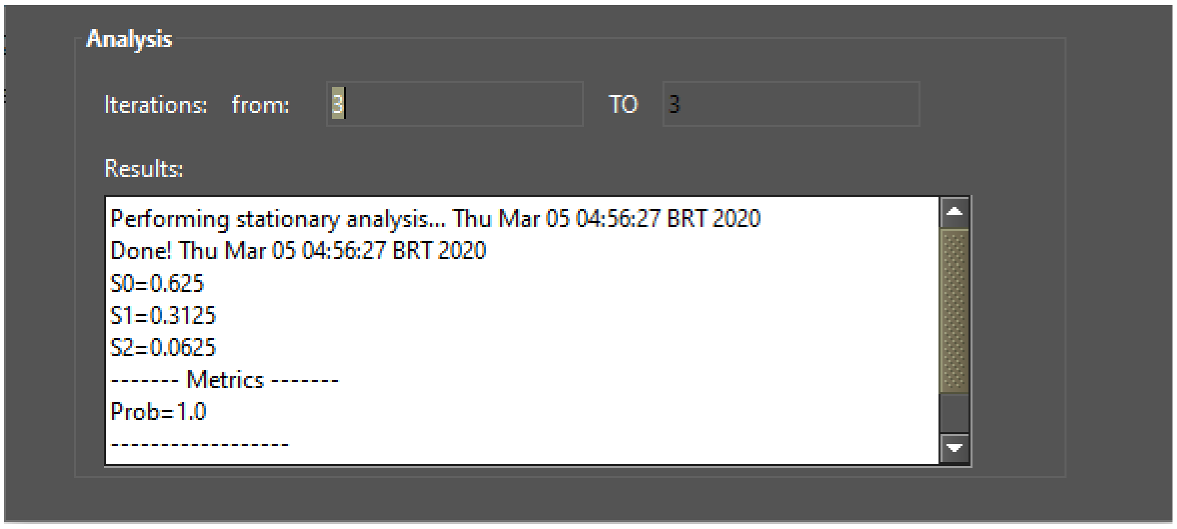
When solving the DTMC through GTH, the user can change the **maximum error** used in the algorithm. The default value for the maximum error is 0.0000001 (10^-7). By clicking on the “Run” button, the solution algorithm is triggered and as soon as it finishes, the results are presented in the text area at the bottom of the window (see Figure 163), and written in a plain text file with the name of the project file appended with the “-StationaryAnalysis.txt” suffix.



Method: Direct - GTH

Error: 0.0000001

Figure 162: Stationary Analysis Window

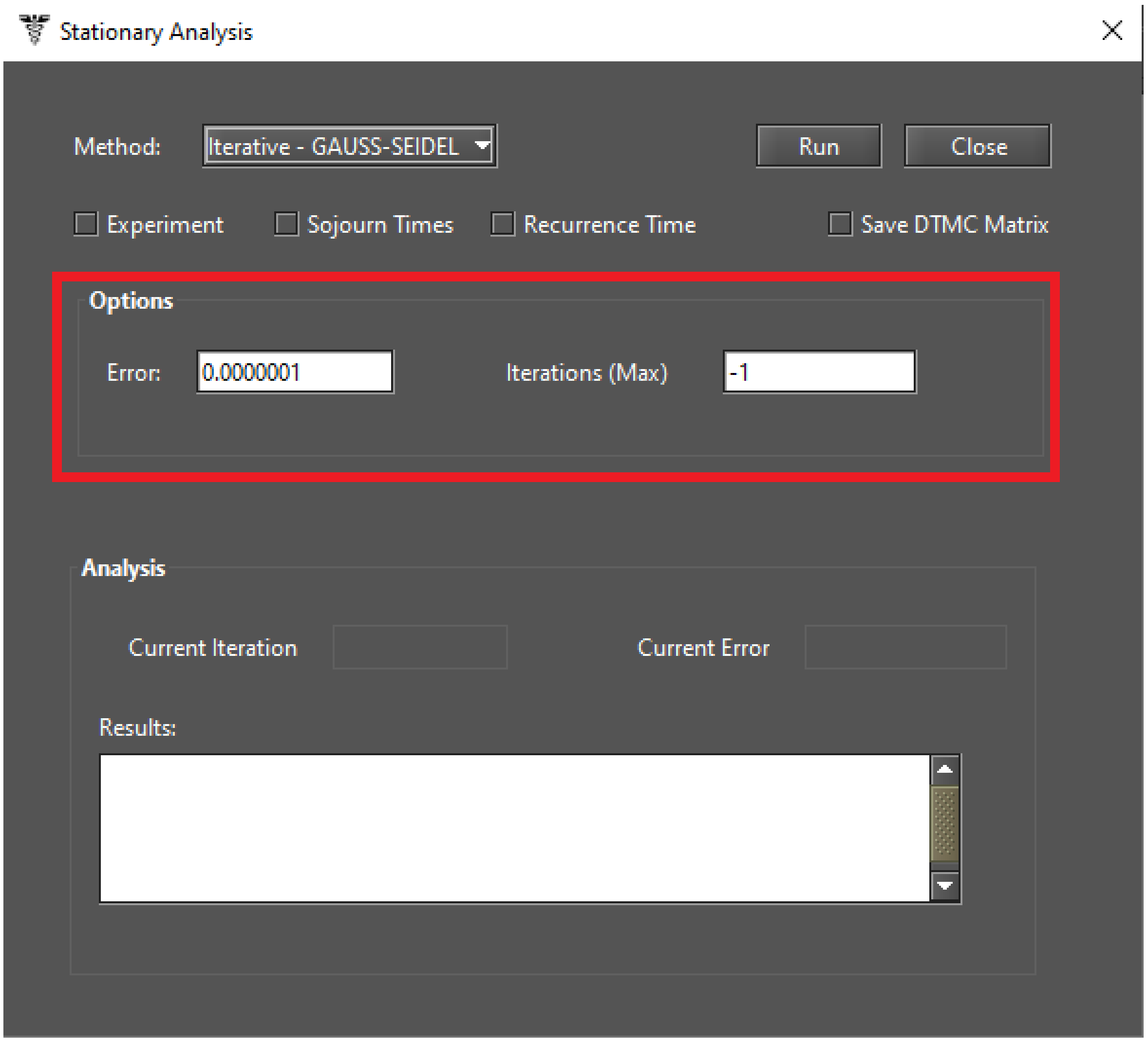


Iterations: from: 3 TO 3

```
Performing stationary analysis... Thu Mar 05 04:56:27 BRT 2020
Done! Thu Mar 05 04:56:27 BRT 2020
S0=0.625
S1=0.3125
S2=0.0625
------- Metrics -------
Prob=1.0
-------------------
```

Figure 163: Stationary Analysis Results

When solving the DTMC through Gauss-Seidel, besides the maximum error, the user can also change the maximum number of iterations. The default value for such a parameter is “-1”, indicating that the algorithm only stops when the convergence of results is reached, considering the entered error (see Figure 164).


Stationary Analysis

Method: Iterative - GAUSS-SEIDEL

Options

Error: 0.0000001

Iterations (Max) -1

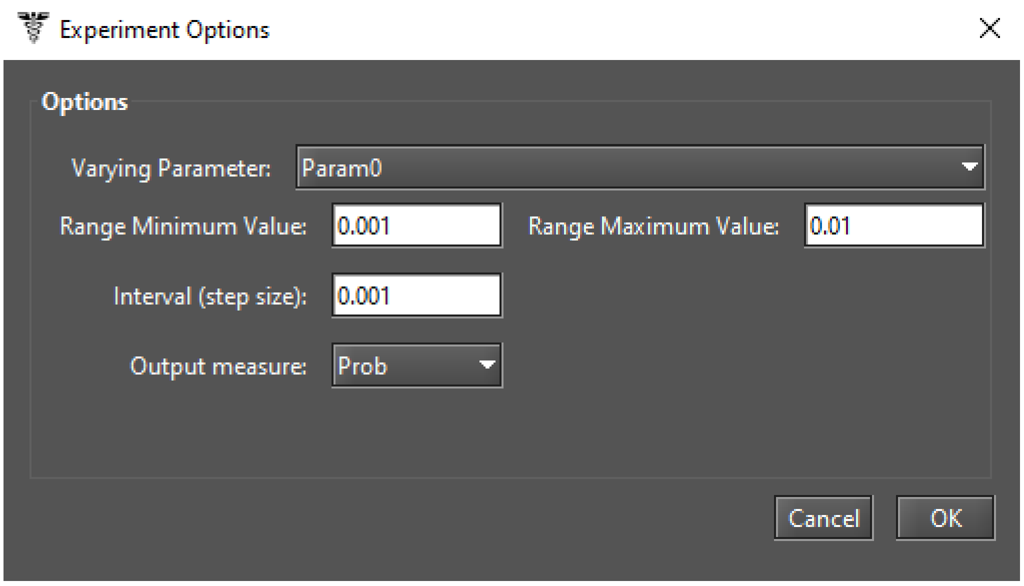
Figure 164: Stationary Analysis Window - Gauss-Seidel Method

The metrics are updated in the DTMC drawing area as soon as the analysis is completed, independently of the chosen solution method.

The DTMC model can also be solved for a range of values of user-defined parameters. This is accomplished by clicking in the “Experiment” checkbox on the “Stationary Analysis” window and then on the “Run” button. A new dialog is displayed in order to define the options for the experiment to be carried out (see Figure 165).

Below, we describe each of them.

- **Varying Parameter.** The parameter to have its value changed.
- **Output measure.** The metric to be evaluated.
- **Range Minimum Value.** The initial value to be assigned to the selected parameter.


Experiment Options

Options

Varying Parameter: Param0

Range Minimum Value: 0.001

Range Maximum Value: 0.01

Interval (step size): 0.001

Output measure: Prob

Figure 165: DTMC Experiment Options

- **Range Maximum Value.** The final value to be assigned to the selected parameter.
- **Interval.** It is the step-size for changing the value of the parameter. The parameter starts with the minimum value and its value is increased by considering the entered interval. At each change, the selected metric is evaluated. The experiment ends when the maximum value for the parameter is reached.

At the end of the experiment, a graph is plotted and the results are shown, as we can see by looking at Figures 166 and 167, respectively.

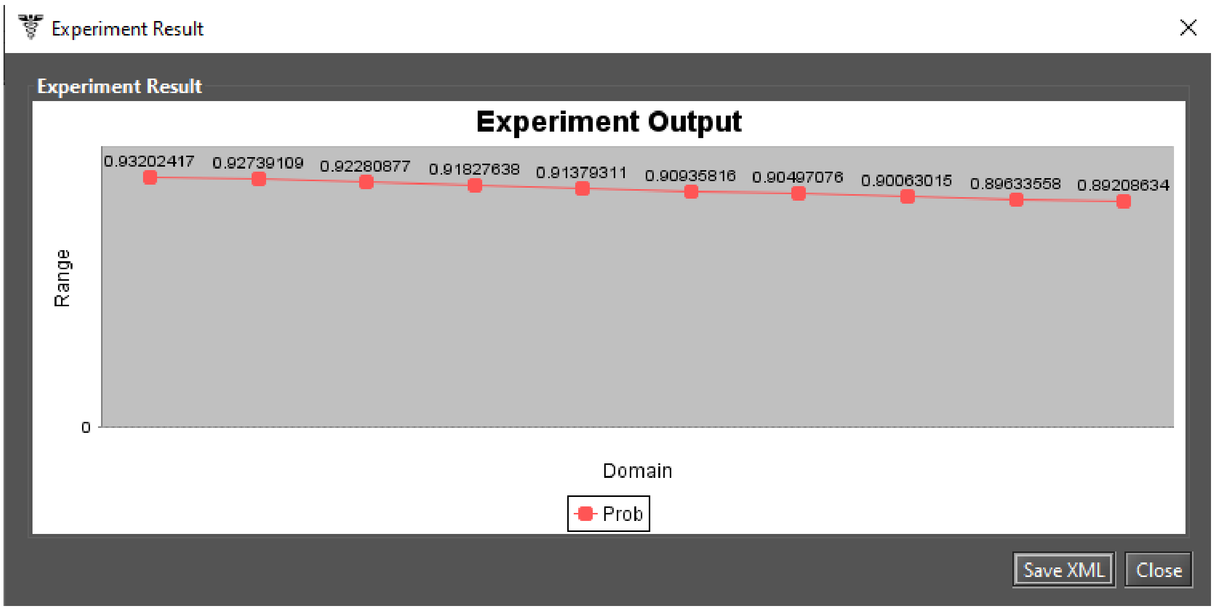
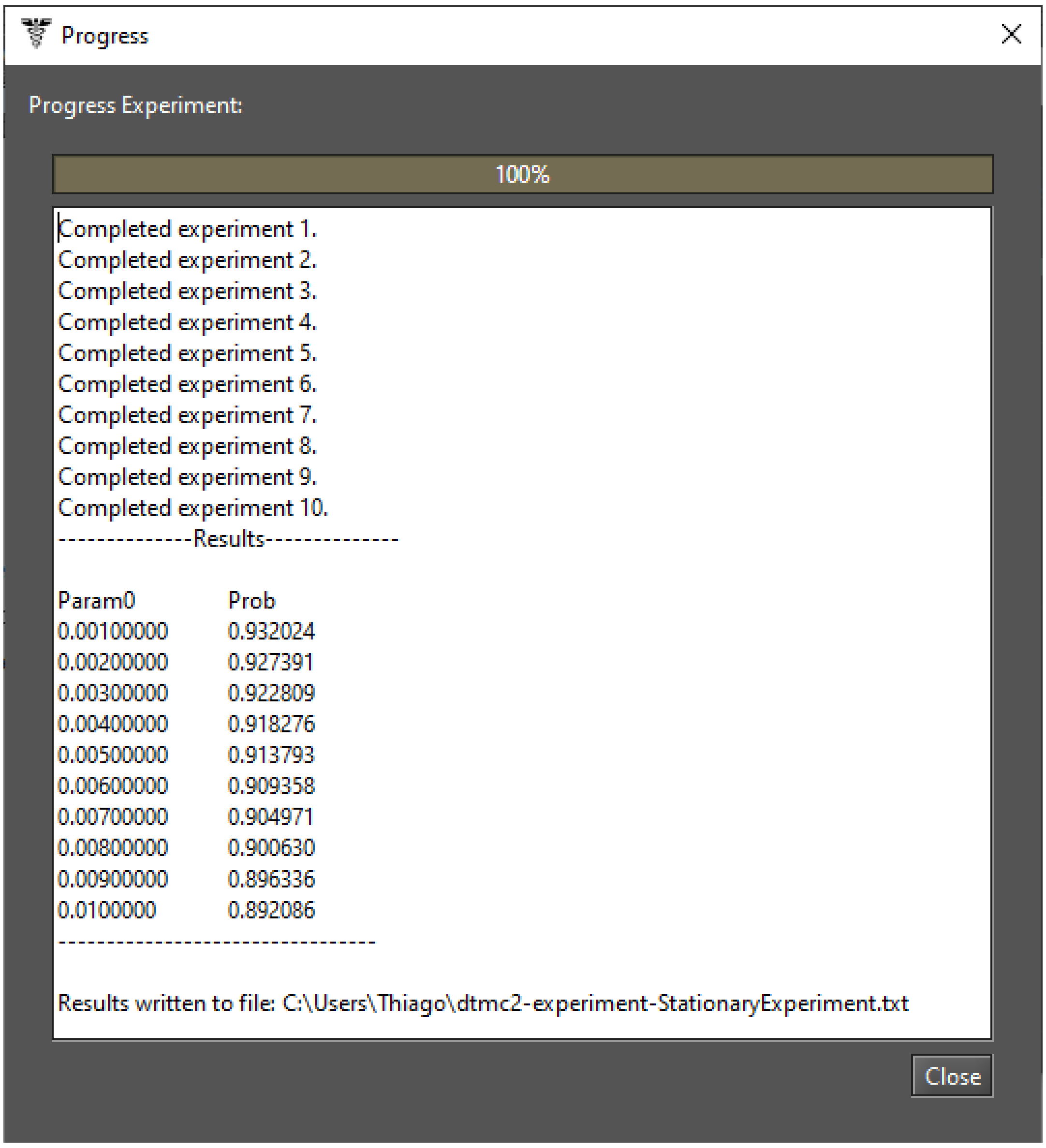


Figure 166: DTMC Experiment Graph



```
Completed experiment 1.
Completed experiment 2.
Completed experiment 3.
Completed experiment 4.
Completed experiment 5.
Completed experiment 6.
Completed experiment 7.
Completed experiment 8.
Completed experiment 9.
Completed experiment 10.
-------------Results-------------

Param0          Prob
0.00100000      0.932024
0.00200000      0.927391
0.00300000      0.922809
0.00400000      0.918276
0.00500000      0.913793
0.00600000      0.909358
0.00700000      0.904971
0.00800000      0.900630
0.00900000      0.896336
0.0100000       0.892086
--------------------------------

Results written to file: C:\Users\Thiago\dtmc2-experiment-StationaryExperiment.txt
```

Figure 167: DTMC Experiment Results

Mercury also computes the following metrics when they are selected in the “Stationary Analysis” dialog:

- **Sojourn times.** Computes the time spent in each state of the DTMC. Listing 5 shows the sojourn times computed for each state of our DTMC (see Figure 159).
- **Recurrence time.** Computes the expected return time for each state of the DTMC. Listing 6 shows the recurrence time computed for each state of our DTMC (see Figure 159).

Listing 5: DTMC Stationary Result - Sojourn Times

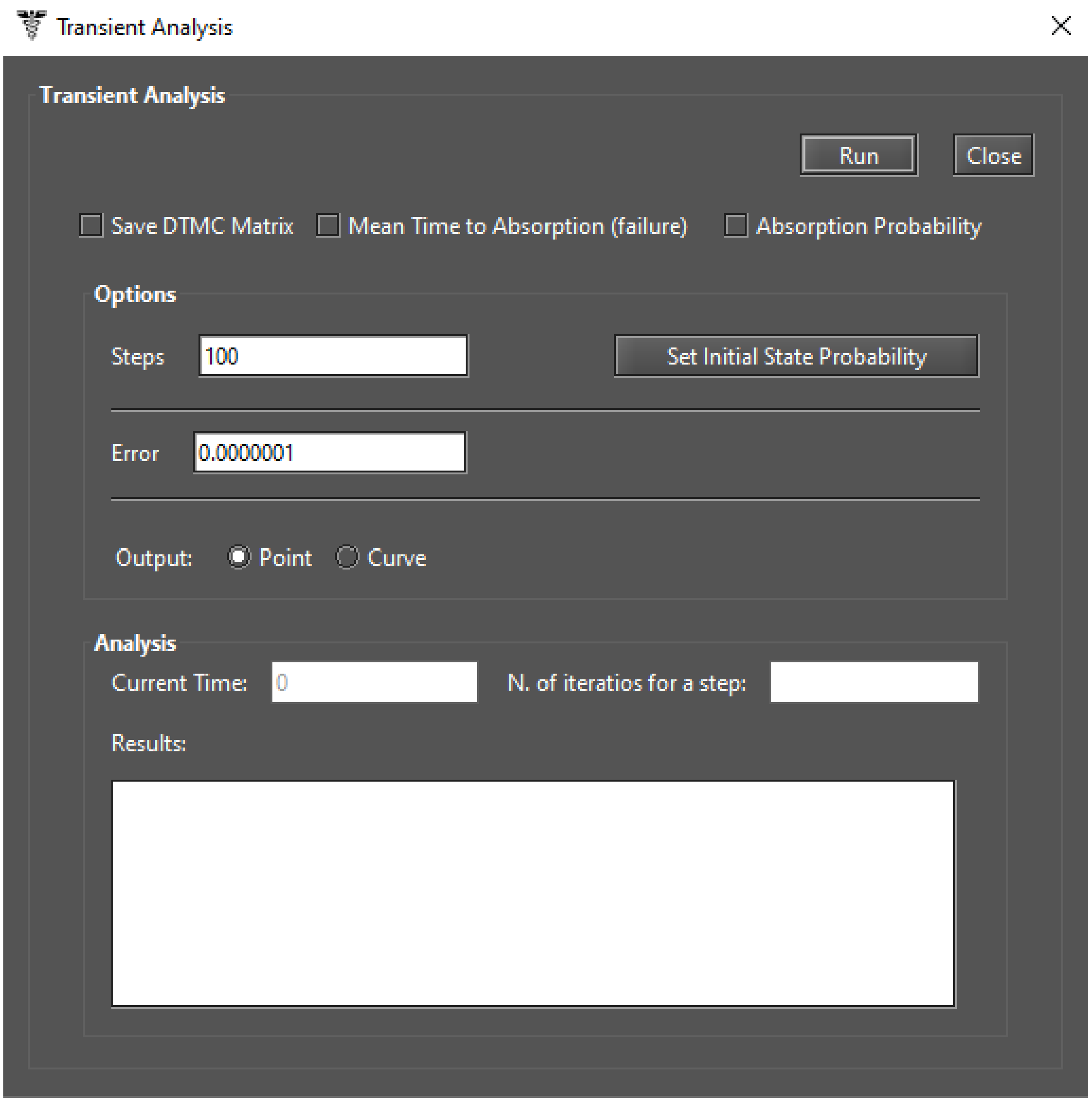
```
Performing stationary analysis... Thu Mar 05 05:49:58 BRT 2020
Done! Thu Mar 05 05:49:58 BRT 2020
S0=0.6586375715496784
S1=0.280391409717778
S2=0.0609710187325436
Hold time: S0=11.494252873563209
Hold time: S1=5.000000000000001
Hold time: S2=2.0
-------- Metrics --------
Prob=0.719608590282222
--------------------
Results written to C:\Users\Thiago\dtmc2-experiment-StationaryAnalysis.txt
```

Listing 6: DTMC Stationary Result - Recurrence Time

```
Performing stationary analysis... Thu Mar 05 05:36:02 BRT 2020
Done! Thu Mar 05 05:36:02 BRT 2020
S0=0.6586375715496784
S1=0.280391409717778
S2=0.0609710187325436
Recurrence time: S0=1.5182857142857142
Recurrence time: S1=3.5664429530201343
Recurrence time: S2=16.40123456790123
-------- Metrics --------
Prob=0.719608590282222
--------------------
Results written to C:\Users\Thiago\dtmc2-experiment-StationaryAnalysis.txt
```

5.3.2 DTMC Transient Analysis

Transient Analysis computes time-dependent probabilities, useful for evaluating the behavior of modeled systems in a particular point in time. Figure 168 shows the “Transient Analysis” window.


Transient Analysis

Transient Analysis

Run Close

☐ Save DTMC Matrix ☐ Mean Time to Absorption (failure) ☐ Absorption Probability

Options

Steps: 100

Set Initial State Probability

Error: 0.0000001

Output: [✓ Point] Curve

Analysis

Current Time: 0

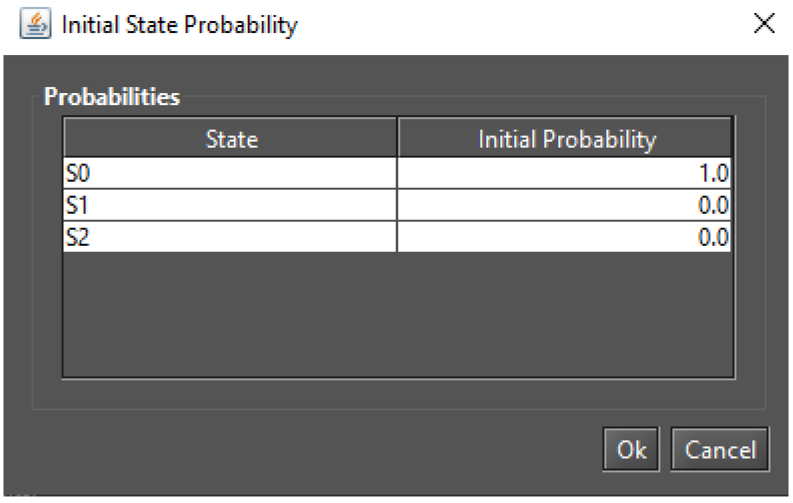
N. of iteratios for a step:

Results:

Figure 168: Transient Analysis Window

When solving the DTMC, the user can define:

- The **steps** for which the analysis will be carried out (default: 100).
- The **error** (default:10^-7). Error to be considered when calculating the results.
- The **initial state probabilities** (default: 1 for the first inserted state, 0 for the remaining states). These probabilities are defined by clicking on the “Set Initial State Probability” button (see Figure 169).



State	Initial Probability
S0	1.0
S1	0.0
S2	0.0

Figure 169: Initial State Probability Dialog

The internal step size will affect the accuracy of results and also the time required for computing the metrics. By clicking on the "Run" button, the solution algorithm is triggered. As soon as it finishes, the results are presented in the text area at the bottom of the "Transient Analysis" window, also they are written in a plain text file having the filename of the project appended with the "-TransientAnalysis.txt" suffix.

The "Transient Analysis" window also allows the user to choose between a **Point** or **Curve** analysis. **Point** analysis is the default option, and it shows the state probabilities only for the specific point in time. **Curve** analysis writes in a plain text file all state probabilities computed from time equals zero until the specified time.

The mean time to absorption (**MTTA**) is a metric that can be computed in the "Transient Analysis" window, by checking the "Mean Time to Absorption (failure)" checkbox. The MTTA value is presented after the state probabilities in the "Results" text area. For MTTA calculation, the user might also define a metric on the drawing area by using the keyword **MTTA** as its expression. The "Absorption Probability" for each state is another metric that is available in the transient evaluation. Listing 7 shows the MTTA and absorption probability by considering an absorbing DTMC as an example.

Listing 7: DTMC Transient Result - MTTA and Absorption Probability

```
Performing transient analysis... Thu Mar 05 06:08:33 BRT 2020
Results written to C:\Users\Thiago\dtmc-transient-TransientAnalysis.txt
Done! Thu Mar 05 06:08:33 BRT 2020
S0=5.275008948446448E-4
S1=0.9994416594620071
S2=3.083964314639397E-5
------- Metrics -------
Prob=5.583405379910387E-4
Absorption probability to state S1: 0.999999999999999
Mean Time to Absorption (MTTA): 14.09395973154361
```

6 EFM Modeling and Evaluation

Energy Flow Model (EFM) is proposed to estimate the sustainability impact and cost of data center architectures without overstepping the energy constraints of each device. This is accomplished with algorithms that traverse the EFM and compute the cost, estimate the environmental impact, and verify the energy flow. The EFM evaluation functionality is responsible for estimating the sustainability impacts of a system (e.g., model) in terms of its lifetime exergy (available energy) consumption. This functionality also computes the total cost that is composed by **initial cost** and **operational cost**. The initial cost represents the budged needed to obtain the system components in order to build the system. The operational cost is the cost to maintain the system in the operational mode.

Figure 170 depicts an EFM model representing a system composed of four components and it demonstrates how the energy flow occurs between them.

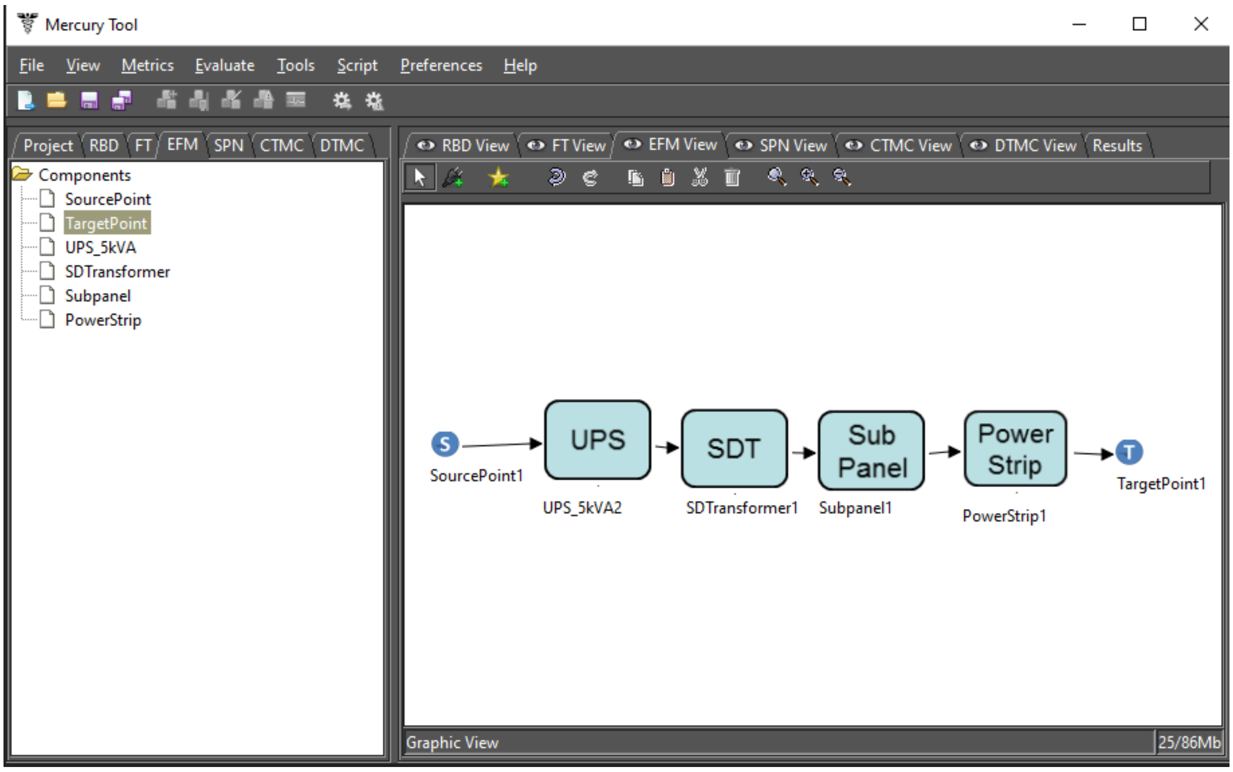


Figure 170: EFM Example

Now we will introduce the EFM toolbar and its available resources. This toolbar is visible when the EFM view is selected. Figure 171 shows the buttons present on it and their descriptions are detailed below.

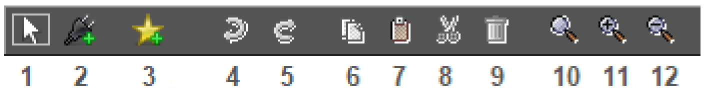


Figure 171: EFM Toolbar

1. **Default Cursor.** Activates the selection mode. This mode makes it possible to select model components in the drawing area.

2. **Insert Component to Canvas.** Add datacenter components to the canvas. The first step to use this functionality is to select the EFM component on the left side panel (see Figure 172). After the selection, the user needs to click on this button and, after that, she needs to click in the drawing area on the location desired to put the new component.

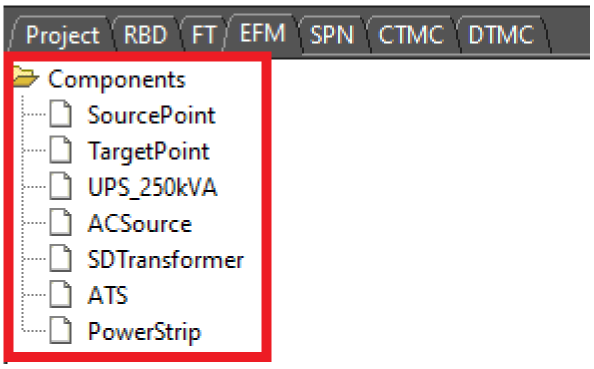


Figure 172: EFM Left-Side Panel

3. **Add a New Component to the EFM Project.** Add new data center components to the project. By clicking on it a dialog appears allowing the modeler to add a component to the project. In order to accomplish this, the user should select the component in this new dialog and confirm the selection by clicking on the “Add” button (see Figure 173). The new component will be available on the left side as shown by Figure 172. It is important to highlight that this function only adds components to the EFM project. The new component is not inserted into the drawing area at this moment. So, in order to do that, it is necessary to click on the “Insert Component to Canvas” button (2), after selecting the desired component on the left side panel.

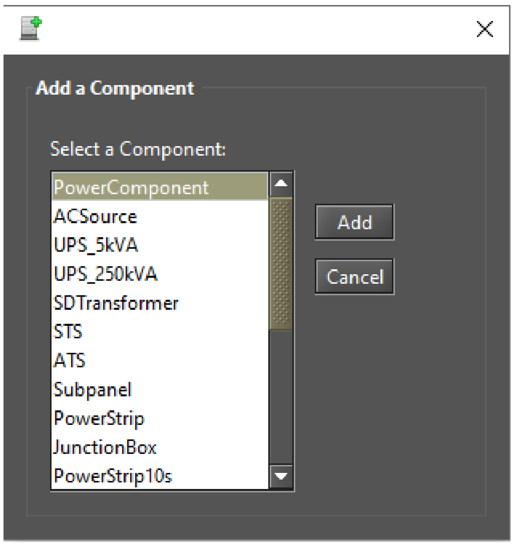


Figure 173: Inserting an EFM Component to the Project

4. **Undo.** Undo the last changes in the drawing area.

5. **Redo.** Redo the last changes in the drawing area.

6. **Copy.** Copy selected model objects to the drawing area.

7. **Paste.** Insert the components currently available on the clipboard into the drawing area.

8. **Cut.** Cut the components selected in the drawing area to the clipboard.

9. **Delete.** Remove the selected components from the drawing area.

10. **Standard Scale.** Apply the standard scale to the drawing area.

11. **Scale Up Image.** Each click scales up the drawing image by 10% percent (zoom in).

12. **Scale Down Image.** Each click scales down the drawing image by 10% percent (zoom out).

Now, we will describe how to perform evaluations on an EFM project by using the Mercury tool. The first step in order to perform the sustainability evaluation is to create an EFM model. To add power or cooling components in the model, users should click on the start icon ("Add New Component to the Project" button). Once clicked on that button, a window appears in which it is possible to select the component to be added to the EFM project (see Figure 173). The list of components on the left side panel is updated after the addition of a component. The next step is to add the selected component on the left side panel to the drawing area as depicted in Figure 174. The user needs to click on the power plug icon on the toolbar ("Add Component to Canvas" button), after that she must click into the drawing area. These last steps need to be performed for each component that composes the evaluated system.

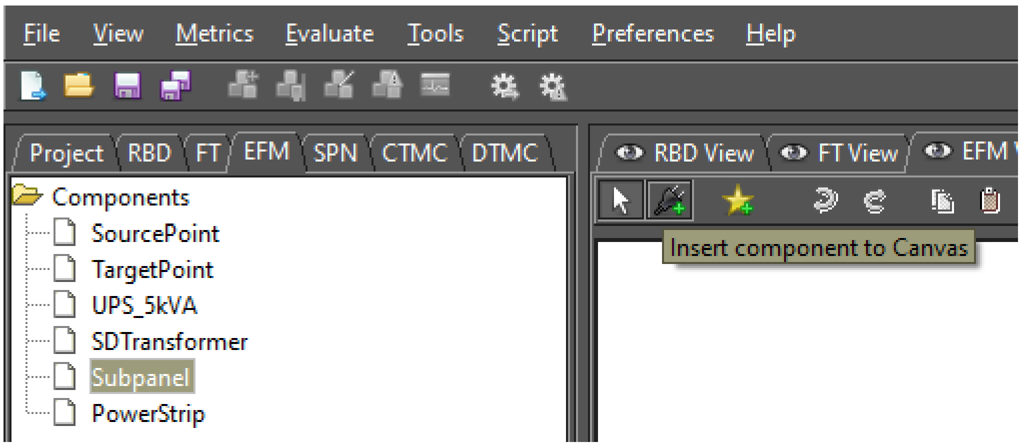


Figure 174: Inserting a Component into the Drawing Area

Once all components are inserted into the drawing area, the user can connect them including SourcePoints and TargetPoint as shown by Figure 175.

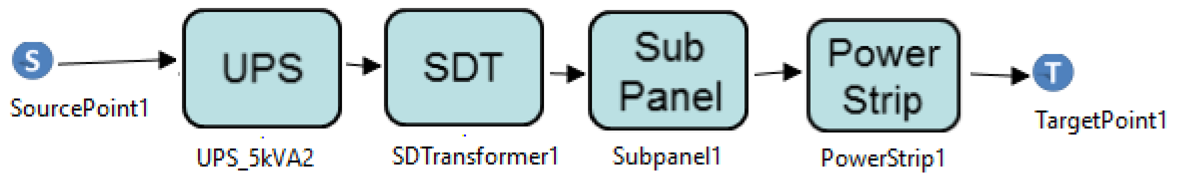
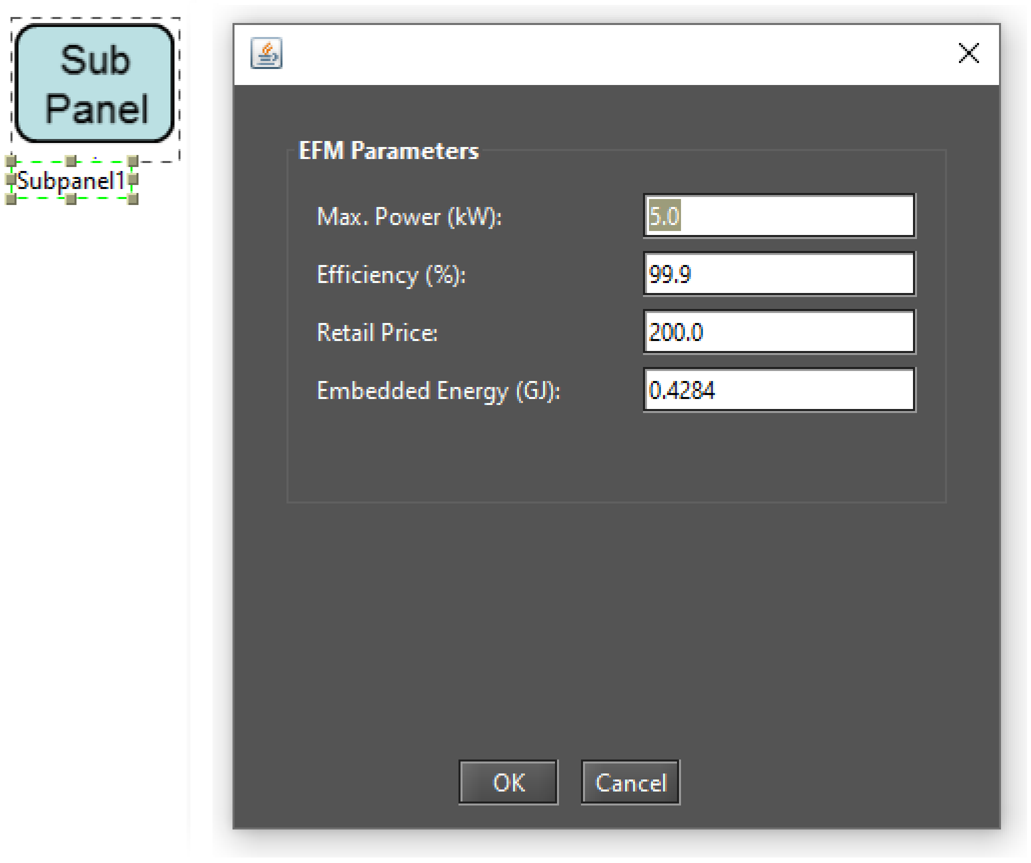


Figure 175: EFM Example

By right-clicking on a component and left-clicking on the “Properties” menu that appears provides a way to edit its properties as shown by Figure 176.



EFM Parameters

Max. Power (kW):	5.0
Efficiency (%):	99.9
Retail Price:	200.0
Embedded Energy (GJ):	0.4284

Figure 176: Component Properties

Additionally, it is important to stress that users have to set the demanded power on the TargetPoint (for power system) or on the SourcePoint (for cooling system). This is done by with a right-click on the Source or Target points and selecting the option “Properties”. After that, a dialog appears and the demanded power may be entered (see Figure 177).

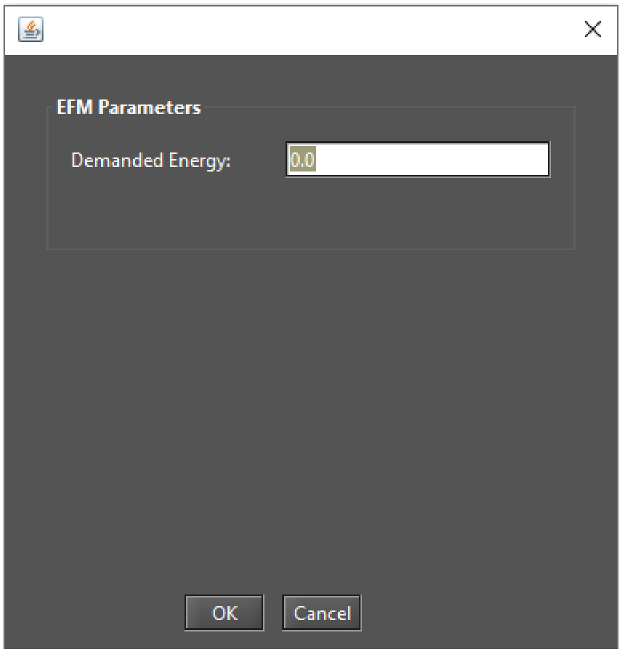


Figure 177: SourcePoint and TargetPoint Property

Once the EFM model is completed, evaluations can be conducted in order to extract some metrics. Evaluations are performed by selecting the desired option on the “EFM Evaluation” menu group available in the “Evaluate” menu on the main menu, as depicted by Figure 178. Five EFM evaluations are available. It is possible to evaluate cost, exergy, energy flow, the last ones combined, and flow optimization.

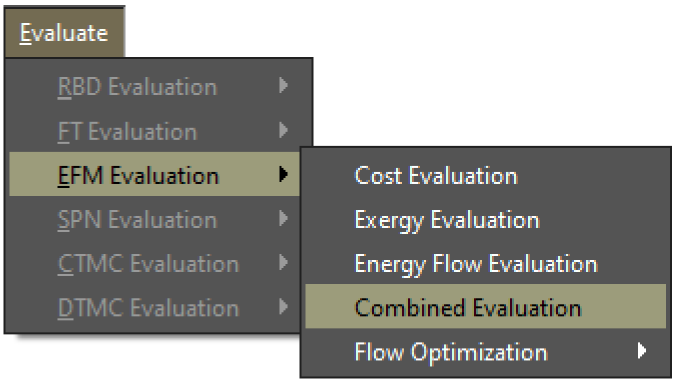


Figure 178: EFM Evaluation Menu

Once selected the combined option, the EFM evaluation of cost, exergy, and energy flow is selected. To conduct those evaluations, users have to provide the EFM parameters depicted in Figure 179. The parameters that the user may provide are availability, period to be considered (in hours), and electricity cost (per kWh).

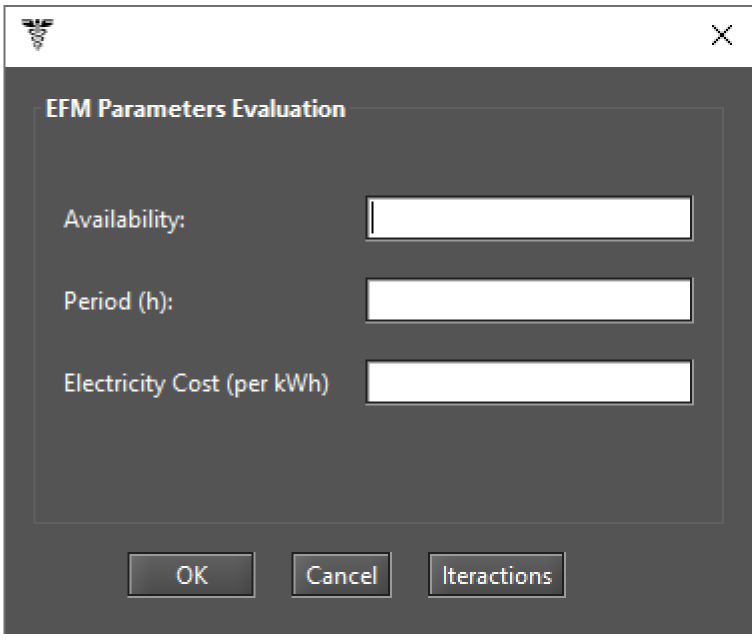
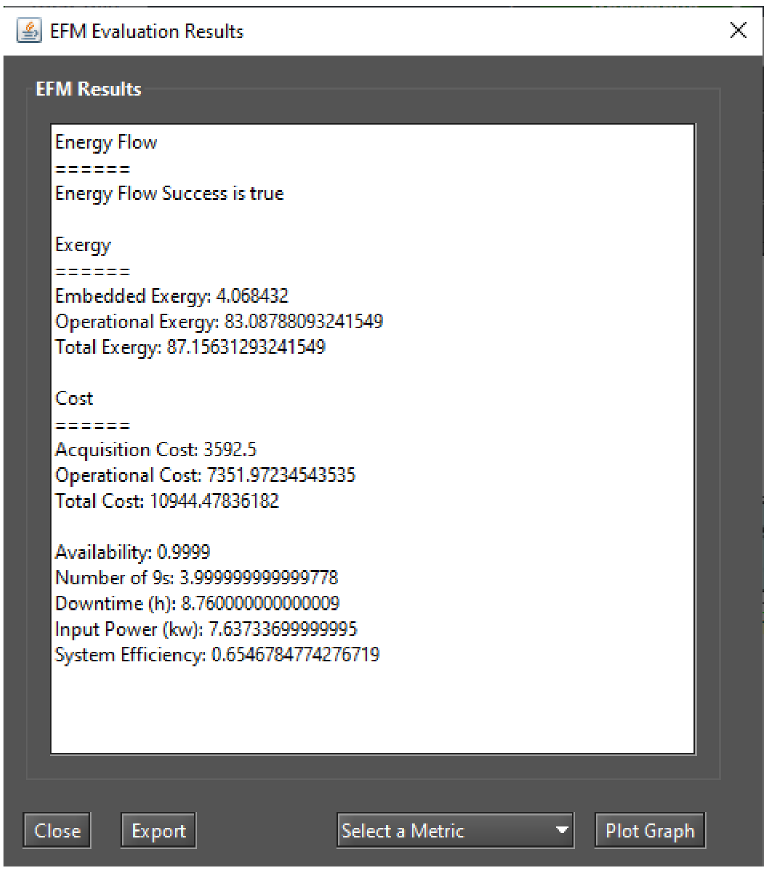


Figure 179: EFM Parameters Evaluation

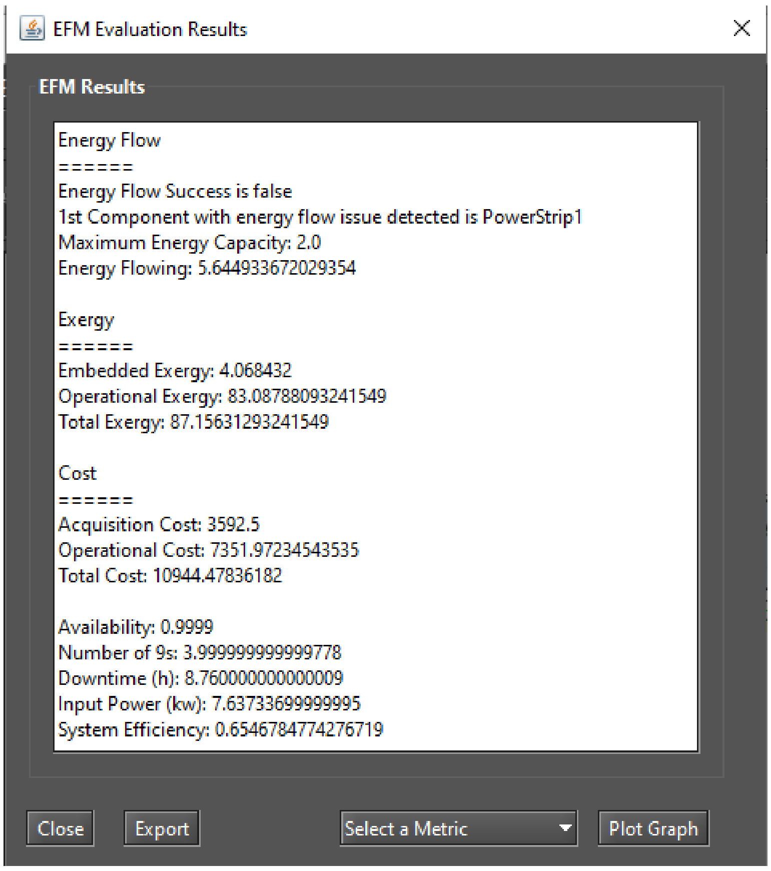
Finally, the result is presented as demonstrated by Figure 180.



```
Energy Flow
======
Energy Flow Success is true

Exergy
======
Embedded Exergy: 4.068432
Operational Exergy: 83.08788093241549
Total Exergy: 87.15631293241549

Cost
======
Acquisition Cost: 3592.5
Operational Cost: 7351.97234543535
Total Cost: 10944.47836182

Availability: 0.9999
Number of 9s: 3.999999999999778
Downtime (h): 8.760000000000009
Input Power (kw): 7.63733699999995
System Efficiency: 0.6546784774276719
```

Figure 180: EFM Results

In this example, the result indicates that the energy flow evaluation returns true meaning that the power constraints present on each device were respected. However, in case this result is false, the Mercury tool shows the component in which the constraint was crossed (see Figure 181). In this window of results, the user has also an option to export the results to a spreadsheet (e.g., a file .xls), or plot a selected metric.



```
Energy Flow
======
Energy Flow Success is false
1st Component with energy flow issue detected is PowerStrip1
Maximum Energy Capacity: 2.0
Energy Flowing: 5.644933672029354

Exergy
======
Embedded Exergy: 4.068432
Operational Exergy: 83.08788093241549
Total Exergy: 87.15631293241549

Cost
======
Acquisition Cost: 3592.5
Operational Cost: 7351.97234543535
Total Cost: 10944.47836182

Availability: 0.9999
Number of 9s: 3.999999999999778
Downtime (h): 8.760000000000009
Input Power (kw): 7.63733699999995
System Efficiency: 0.6546784774276719
```

Figure 181: EFM Results with the Energy Flow Evaluation False

6.1 Power Load Distribution Algorithm - PLDA

A Power Load Distribution Algorithm (PLDA) is proposed to minimize the electrical energy consumption of the EFM models [7]. The PLDA is based on the Ford-Fulkerson algorithm, which computes the maximum possible flow in a flow network [8]. The network is represented by a graph, where the transport capacity of the devices is defined in the edges. The algorithm begins by traversing the graph, searching for the best flows between two specific points in the graph. If a particular path lacks the capacity to support all of the flow demanded, then the residual flow is redirected to other paths. The Priority First Search (PFS) is the adopted method for selecting the path between the nodes [9, 10]. The PFS chooses the path according to the highest electrical capacities of nodes in the graph [11]. Figure 182 shows how to call the PLDA function.

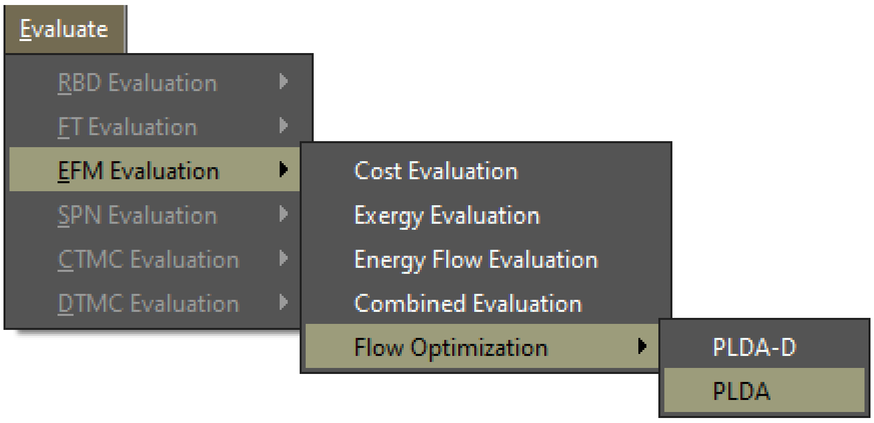
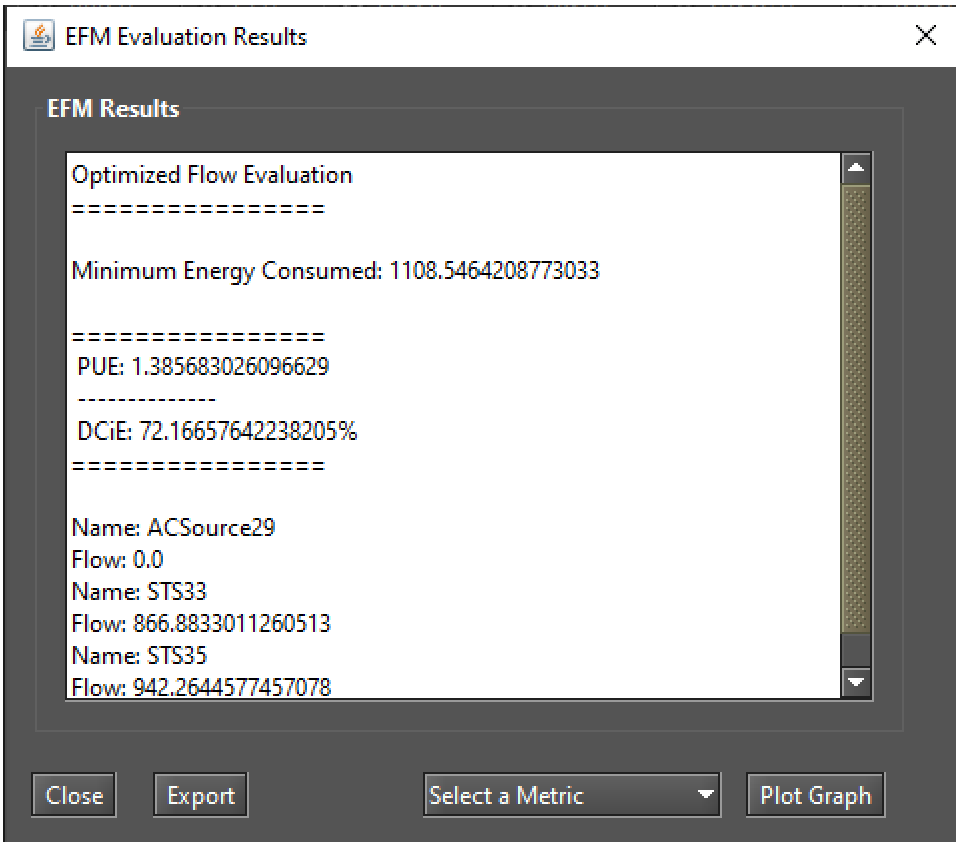


Figure 182: PLDA Optimization Function

Figure 183 depicts the results of the PLDA algorithm, with the minimum energy consumed, PUE, and DCiE highlighted.



```
Optimized Flow Evaluation
================

Minimum Energy Consumed: 1108.5464208773033

================
 PUE: 1.385683026096629
 --------------
 DCiE: 72.16657642238205%
================

Name: ACSource29
Flow: 0.0
Name: STS33
Flow: 866.8833011260513
Name: STS35
Flow: 942.2644577457078
```

Figure 183: PLDA Optimization Results

6.1.1 Example of PLDA execution

Figure 184 illustrates the EFM model of a specified architecture. In the example, all the edge weights are set to the default value, one. The power flow is computed by traversing the graph from the target to the source node.

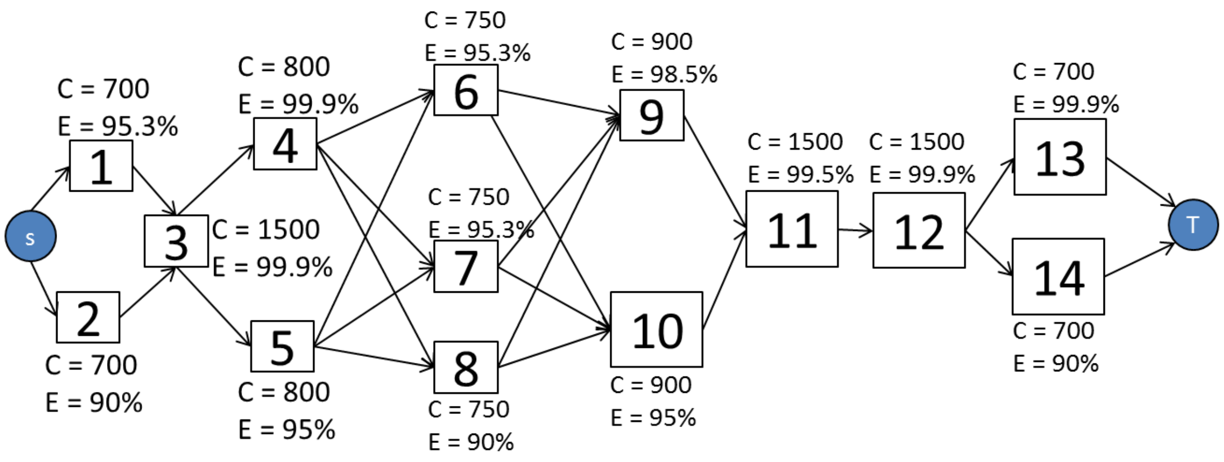


Figure 184: EFM Model

Figure 185 depicts the EFM model after the execution of the PLDA. It should be noted that the weights on the edges have changed, optimizing the power flow through a best weights distribution.

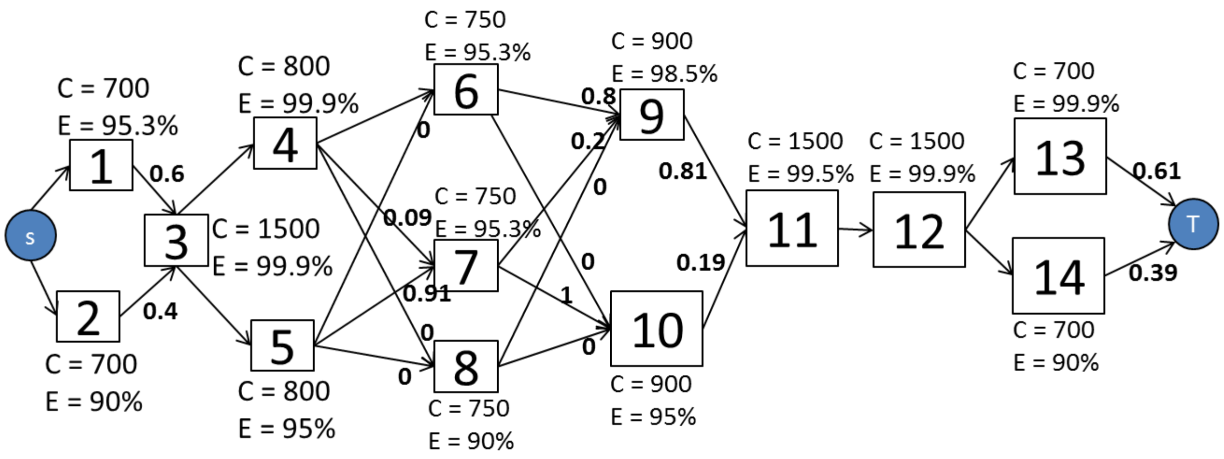


Figure 185: EFM Model After PLDA Execution

Table 1 presents a summary of the results obtained by the PLDA. Column "*Improvement*" depicts the improvement. The system efficiency is improved by over 4.2%; consequently, the associated cost and sustainability figures are improved by 4.2% and 20.4%, respectively. Availability results were also computed with RBD/SPN models, but are not included here.

Table 1: Summary Results Before and After PLDA Execution

Metric	**Before**	**After**	**Improvement (%)**
Availability (%)	0.99999226	0.99999226	0
Number of 9 s	5.111	5.111	0
Downtime (hs)	0.0677	0.0677	0
Input Power (kW)	1,312.63	1,259.64	4.2
System Efficiency (%)	76.18	79.38	4.2
Operational Cost (USD)	1,264,849	1,213,784	4.2
Operational Exergy (GJ)	9,859.32	8,188.11	20.4

6.2 Power Load Distribution Algorithm in Depth search (PLDA-D)

A Power Load Distribution Algorithm - Depth (PLDA-D) is proposed to minimize the electrical energy consumption of the EFM models [7]. It is an evolution of the PLDA algorithm (see Section 6.1), applies for the same problem but with a big difference in the technique of graph search. In the PLDA-D the model EFM is searched in-depth, choosing always the best path in a depth search to distribute the weights of the edges. The PLDA-D is based in the Bellman [12] and Ford and Fulkerson [8] flow algorithm, but with many adaptations. The PLDA-D is divided into three phases: initialize, kernel, and the search for the best path. Figure 186 demonstrates how to call the PLDA-D function. Figure 187 depicts the results generated by applying the PLDA-D algorithm, with the minimum energy consumed, PUE, and DCiE highlighted.

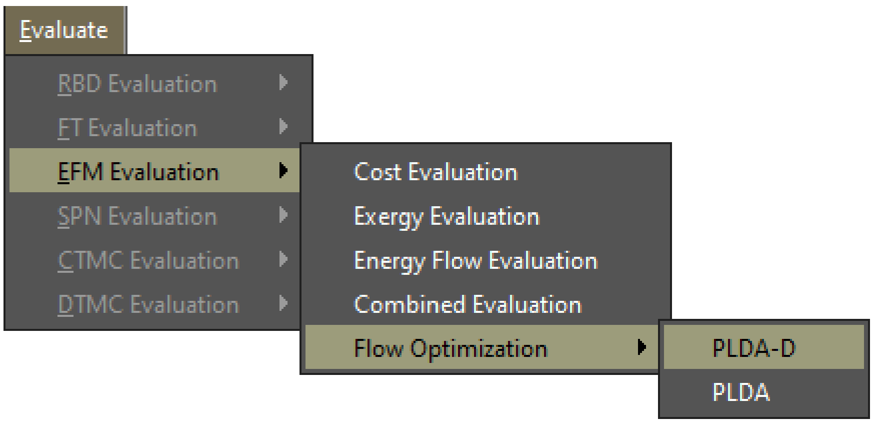
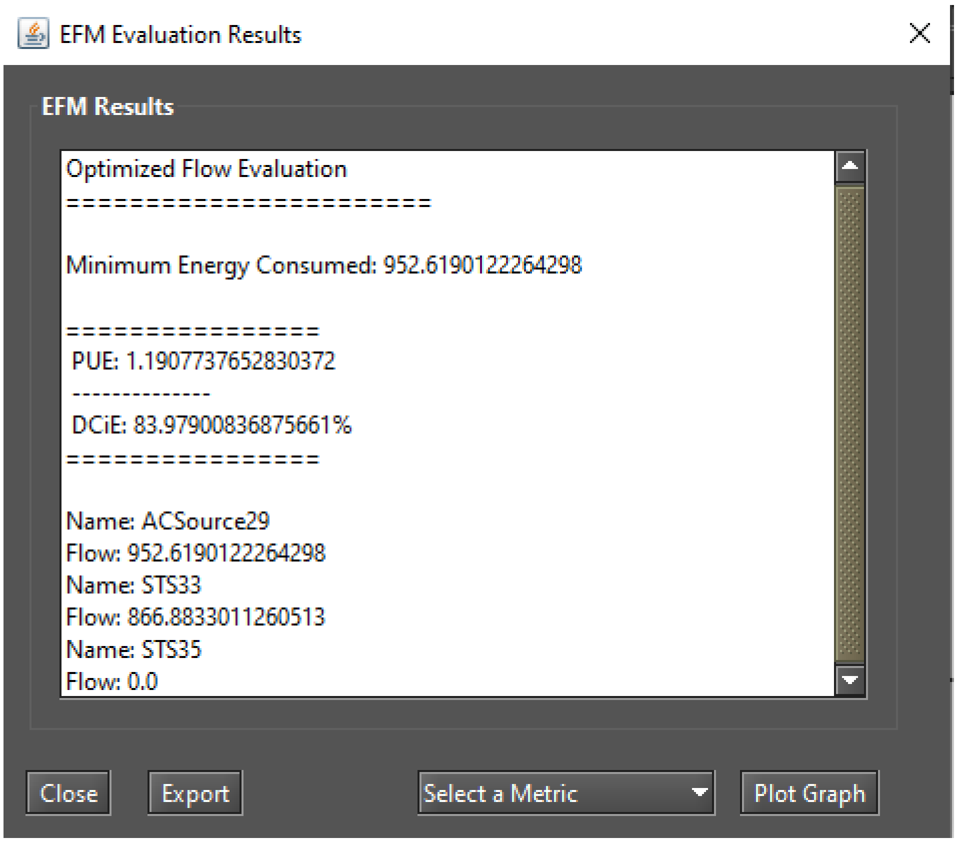


Figure 186: PLDA-D Optimization Function



```
Optimized Flow Evaluation
=======================

Minimum Energy Consumed: 952.6190122264298

================
 PUE: 1.1907737652830372
 -------------
 DCiE: 83.97900836875661%
================

Name: ACSource29
Flow: 952.6190122264298
Name: STS33
Flow: 866.8833011260513
Name: STS35
Flow: 0.0
```

Figure 187: Results for Applying the PLDA-D Optimization

6.2.1 Example of PLDA-D Execution

Figure 188 illustrates the step-by-step of the PLDAD execution, highlighting some variables, edges, and weights. Lets consider the model represent by Figure 188.a with three electrical components A, B, C, each one with efficiency 80, 90 and 95 % respectively. This means that if a component has efficiency of 90%, 10% of the energy that passes through it is lost. The other symbols of the model are S for the node $Source$, with can be represented by an electrical utility and T, for the node $Target$, with can be represented by a computer room.

In the example, the demand (Dem) and efficiency (Ef) values are known. The value of the $Target$ node Acc is set to one. The others accumulated costs (Acc) are set to zero and the edge weights are set to the default value one, respectively, as depicted in Figure 188.a, a perfect representation of an EFM model. The phase one of the PLDAD algorithm is represented by the Figure 188.b when all the variables are initialized in all vertices. Actual cost ($ActCost$) to infinite, child to null ($Child$) and accumulated cost to zero (Acc).

Phase two starts in the Figure 188.c following to the Figure 188.h. At this stage, the best path is selected according to the efficiency of each component, through a scan in-depth and respecting the limits of capacity of each equipment. In Figure 188.c the values of the $ActualCost$ and $AccumulatedCost$ are computed and the best child is chosen, according to the lower value of the variable $ActCost$. This value is used to select the best child for a given node.

A very important step in this phase is represented by Figure 188.g. After the calculations of the variables Acc and $ActCost$, it was verified that the $ActCost$ for the current path (3.39) was less than the $ActCost$ of the

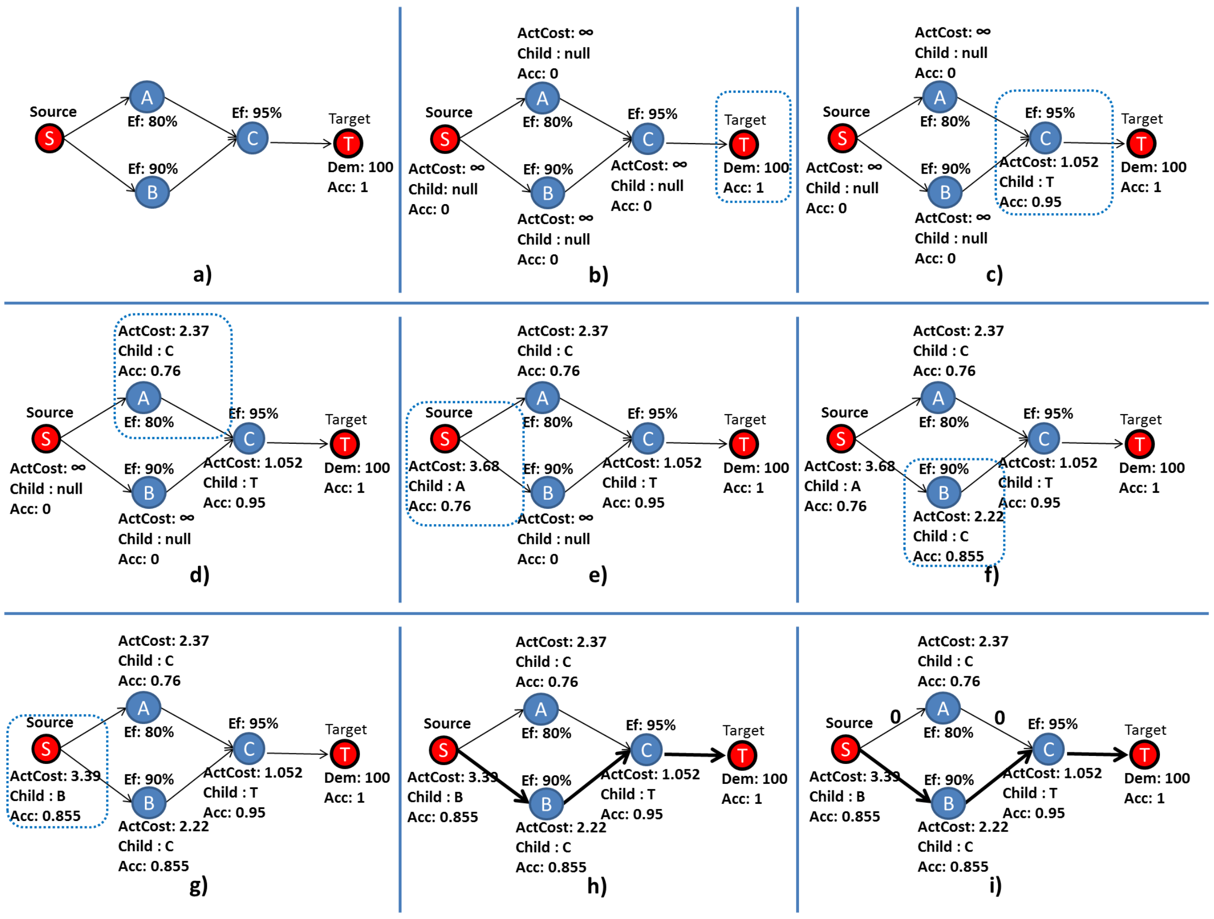


Figure 188: Exemple PLDAD Execution

previous path (3.68) to get to the *Source* node. Thus, the *Source* node has a change in the values of its variables and the best *Child* now is the node *B* and not more node *A*. In other words, it is less costly to get to the source node for B by A, so B represents a better path than A.

Figure 188.h represents the end of phase three. For this example, the best path from note Target to Source is: *Target*, *C*, *B*, *Source*. In Figure 188.i the flow is distributed, according to the weights of the edges. With these values, the EFM computes the minimum possible value for the input power, reducing all the values associated with data center power consumption.

7 FT Modeling and Evaluation

Fault Trees (FTs) and RBDs differ from each other in their final purpose. FT is a top-down logical diagram and it makes possible to create a visual representation of a system that shows the logical relationships between the associated events and causes lead that lead the system to failure. When creating a new project, the default FT model is created with an empty top event. In the FT view, the default model presents a top event named **FAILURE**. This top event has the word **underfined** as a description since no event leads to it, as shown by Figure 189.

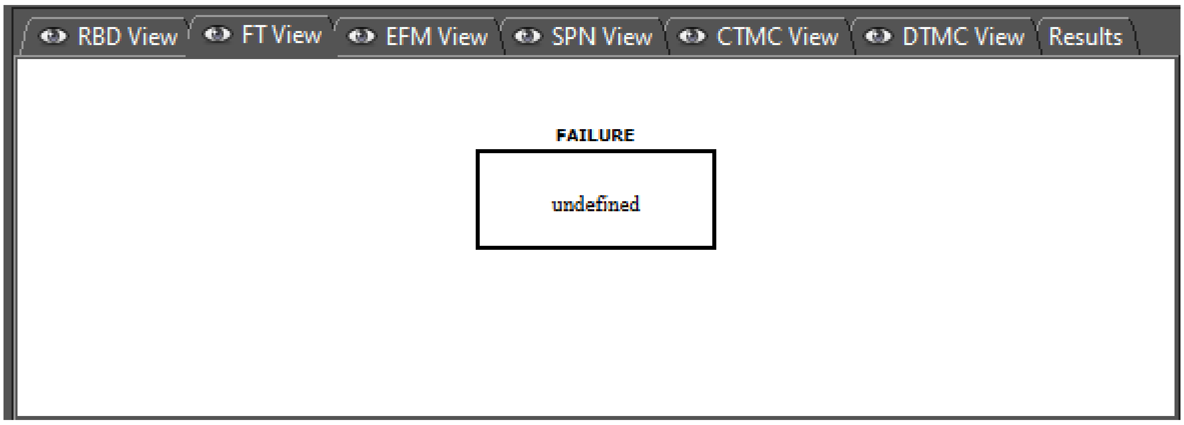


Figure 189: Initial FT Model

By using the Mercury tool we can handle two types of nodes: basic events and gates (logic ports). Basic events are represented as leaf nodes, as depicted by Figure 190. On the other hand, each supported gate has its own graphical representation. As we can see by looking at Figure 191, Mercury supports three types of gates: AND, OR, and K-out-of-N (KooN).

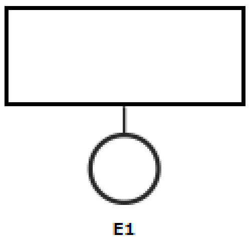


Figure 190: Basic Event

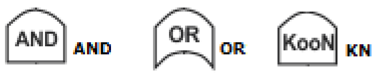


Figure 191: Types of Gates Supported by the Mercury Tool

The events leading to the top-event **FAILURE** must be directly linked to a **GATE**, making it possible to evaluate the probability of an event happening based on the probability obtained by joining basic events and child gates.

In order to create a gate, the user must right-click on the **FAILURE** event. On the pop-up menu that appears there is only one menu-item available: “Add Gate”. Figure 192 demonstrates how to add a gate to the top-event. Users can choose between three different gates: *AND*, *OR*, and *KooN*.

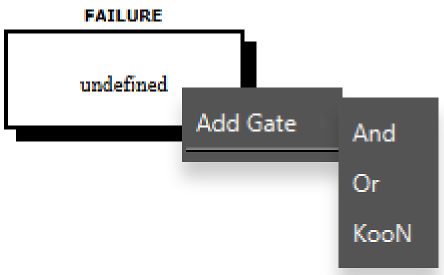
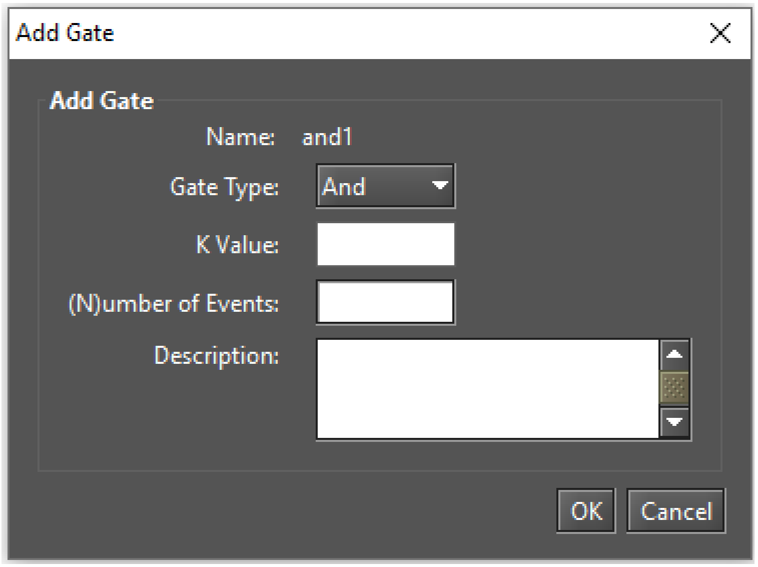


Figure 192: Adding New Gates

After choosing the type of gate to be inserted, the “Add Gate” dialog box appears (see Figure 193). Below, we describe each field of this dialog box.



Name: and1

Gate Type: And

K Value:

(N)umber of Events:

Description:

Figure 193: Add Gate Window

- **Name.** The name of the component, which cannot be changed. This name is automatically generated by the tool.
- **Gate Type.** The type of the gate to be inserted. The user can change the type, if necessary .
- **K Value.** When inserting a KooN gate, this field becomes enabled. A KooN gate represents a set of identical components/events in a single node. It represents how many events (at least) (K) must occur for a failure to occur, considering the (N)umber of events entered (K-out-of-N). Figure 194 shows a KooN gate and the values for the parameters K and N are presented in the gate description.

In the current version of the Mercury tool, it is not possible to add child gates to a KooN gate. In the next version, we intend to overcome this limitation.

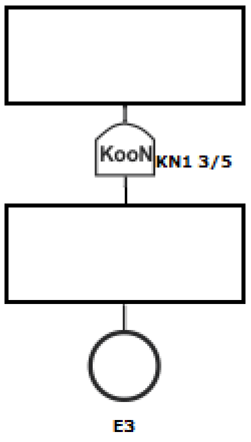


Figure 194: KooN Gate

- **(N)umber of Events.** When adding a new gate, it is necessary to enter the number of basic events to be inserted into the gate. Each gate must have at least two nodes. After insertion, a basic event can be replaced with a gate. Thus, making it possible to define an FT model having a large number of levels.

- **Description.** A description to be attached to the gate. This description is shown in the rectangle at the top of the gate.

After entering the required fields and by clicking on the “OK” button, Mercury inserts the gate and basic events and updates the entire FT graph. As an example, we inserted an AND gate having two basic events, as depicted by Figure 195.

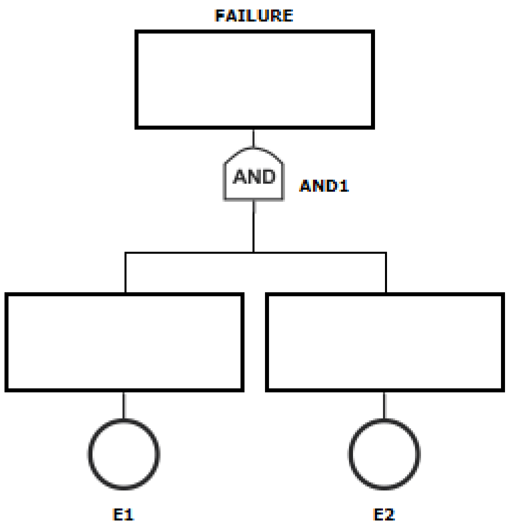
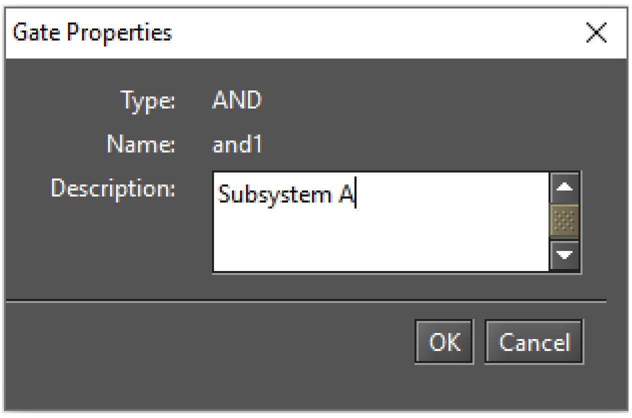


Figure 195: AND Gate with Two Basic Events

A double left click on an gate shows the “Gate Properties” dialog (see Figure 196).



Type: AND
Name: and1
Description: Subsystem A

Figure 196: Gate Properties Dialog

By changing the description of the gate and confirming this change, the model is updated. Figure 197 shows the graph updated.

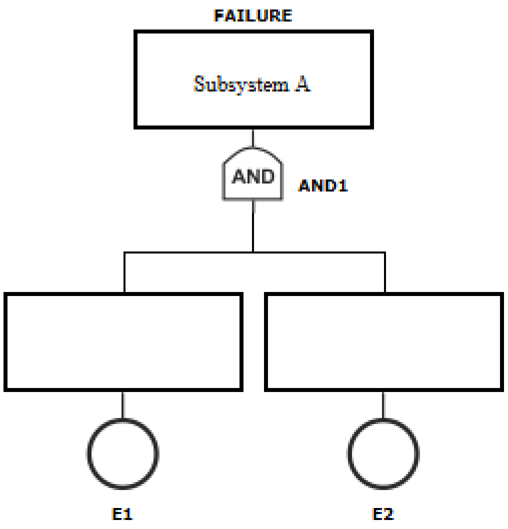


Figure 197: Updating the Gate Description

Another way to edit the properties of a gate is to right-click on it and choose the “Properties” option from the menu that appears. As we can see by looking at Figure 198, some options are available.

Below, we describe the options available in this popup menu.

- **Add events.** When selecting it, a dialog box appears (see Figure 199). In this window, the user must enter the number of events to be inserted into the selected gate. After that, it is necessary to define the properties for each basic event inserted.

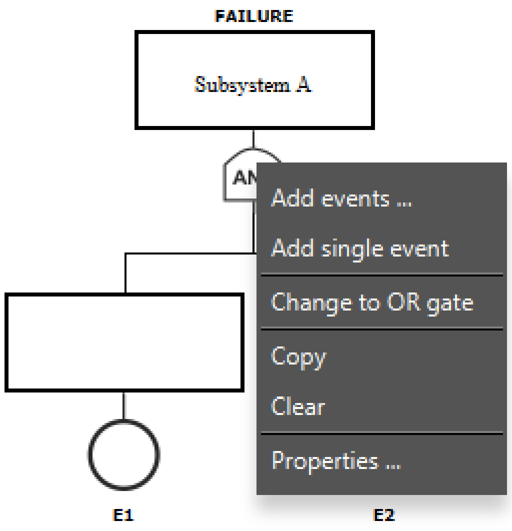


Figure 198: Gate Popup Menu

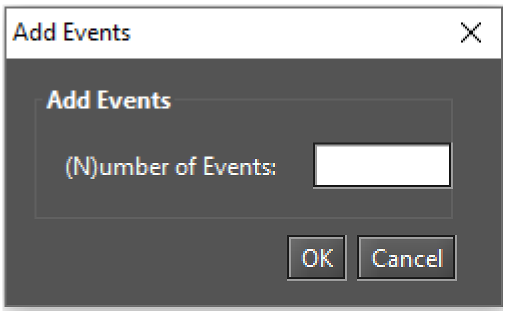


Figure 199: Add Events Dialog

- **Add single event.** Insert an empty event into the model. By considering our example (see Figure 197), after selecting “Add single event” from the popup menu that appears for the AND gate, a basic event is inserted into this gate, as we can see highlighted in Figure 200.

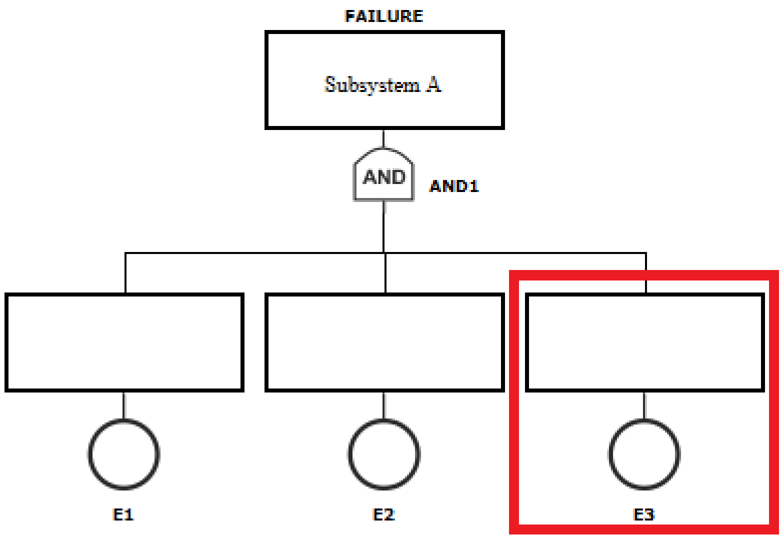


Figure 200: AND Gate with a New Basic Event

- **Change to {OR, GATE} gate.** Change the type of the selected gate. When a port logic OR is selected, the only available option is to change it with an AND gate. The opposite applies for an AND gate. As an example, by considering our model, when selecting this option, the Mercury tool changes the AND gate (see Figure 200) with an OR gate (see Figure 201).

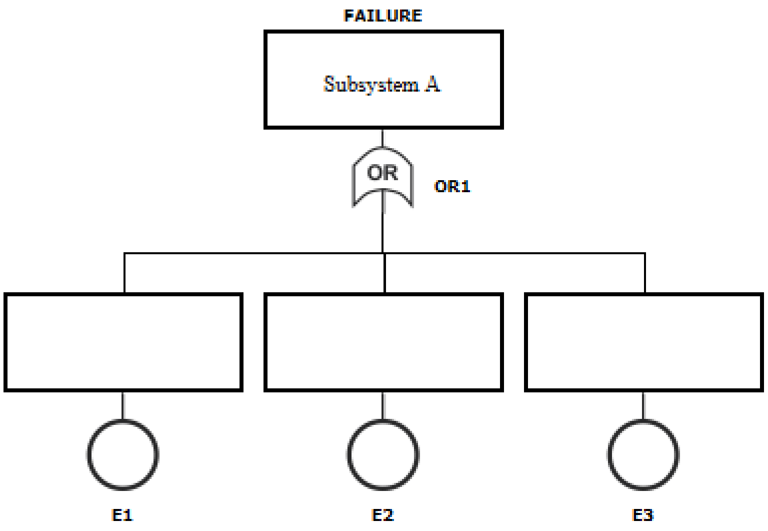
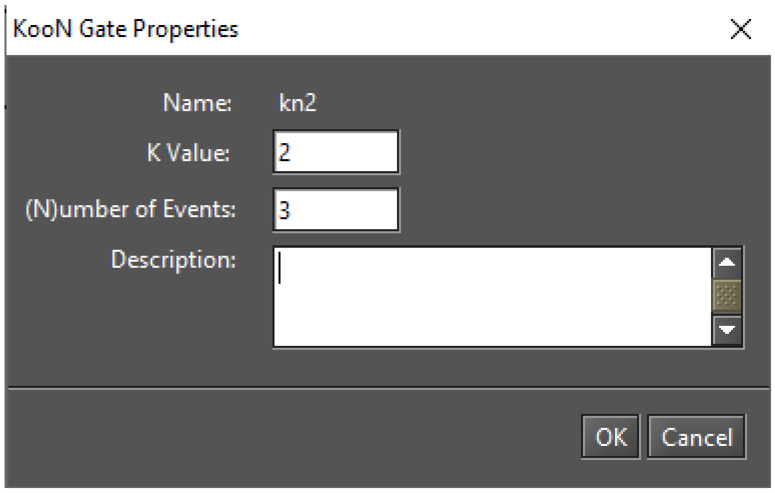


Figure 201: Changing the Gate Type

- **Copy.** Copy the selected gate and all its child components to the clipboard.
- **Clear.** Empty the model. This option is only available when the selected gate is the top-level gate.
- **Properties.** Make it possible to change the properties of the selected gate. Figure 196 shows the "Gate Properties" dialog for AND and OR gates. For a KooN gate, when accessing its properties, besides the description, it is also possible to change the parameters K and N, as we can see by looking at Figure 202.


KooN Gate Properties

Name: kn2

K Value: 2

(N)umber of Events: 3

Description:

Figure 202: KooN Gate Properties Dialog

By right-clicking on a non-top gate, a slightly different popup menu is displayed, as we can see by looking at Figure 203.

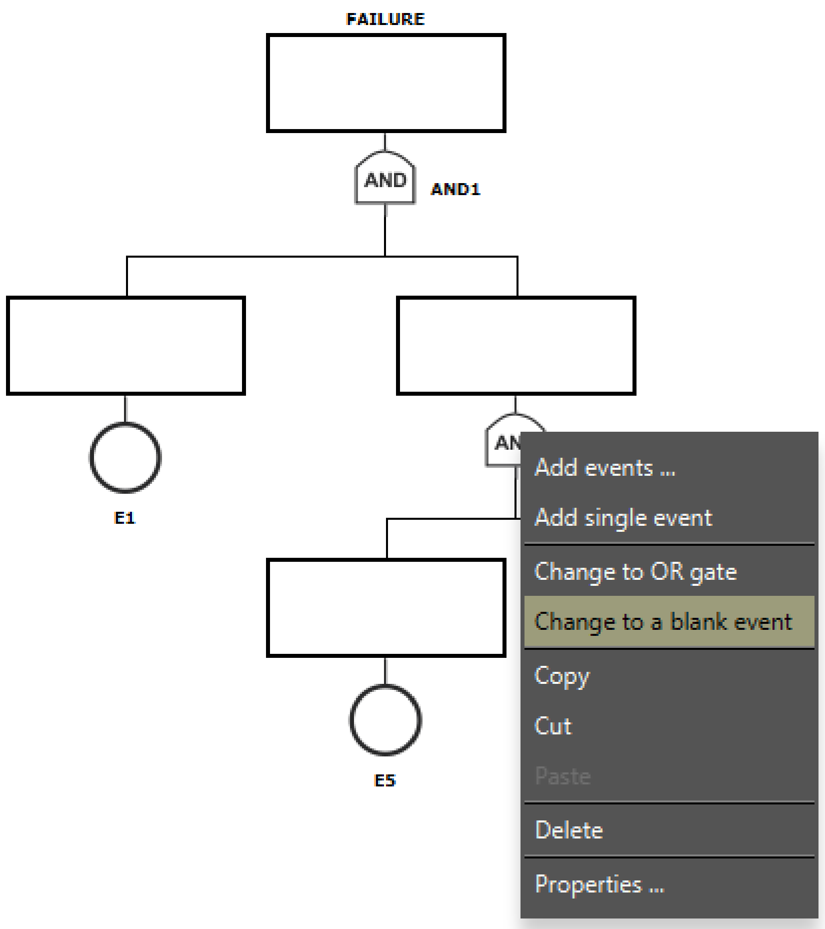


Figure 203: Popup Menu for Non-Top Level Gates

Below, we present the options that we have not yet presented and that are available for non-top gates.

- **Change to a blank event.** Replace the gate and all its child nodes, if any, with an empty event.
- **Cut.** Cut the selected gate to the clipboard.
- **Paste.** Replace the selected gate with the component in the clipboard. This option is only enabled when a component has been copied to the clipboard.
- **Remove.** Remove the selected gate and all its child nodes.

.

By right-clicking on any basic event, a pop-up menu appears, as we can see by looking at Figure 204.

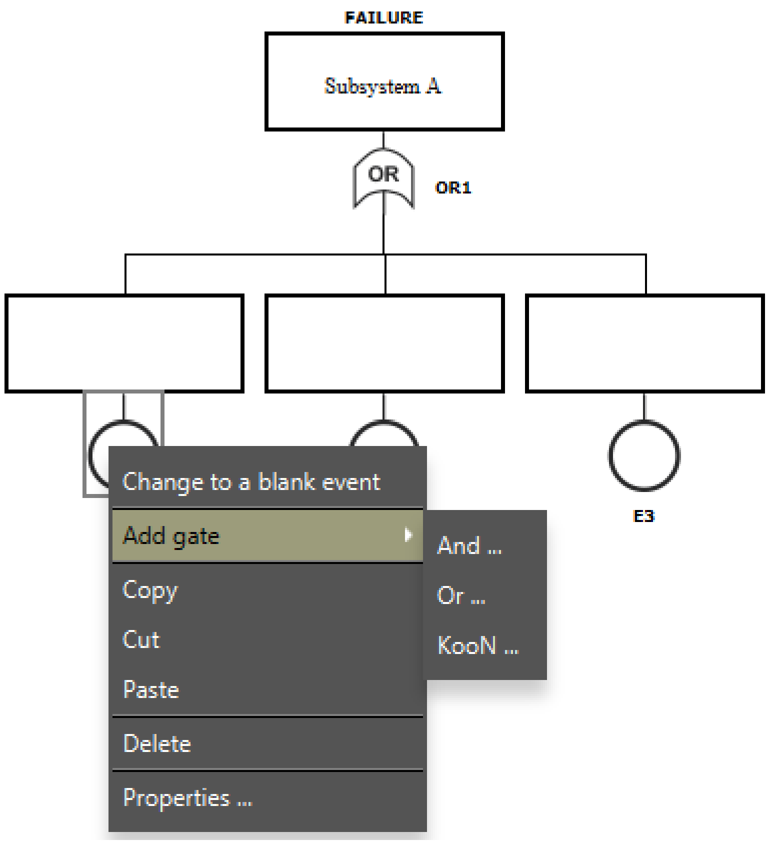
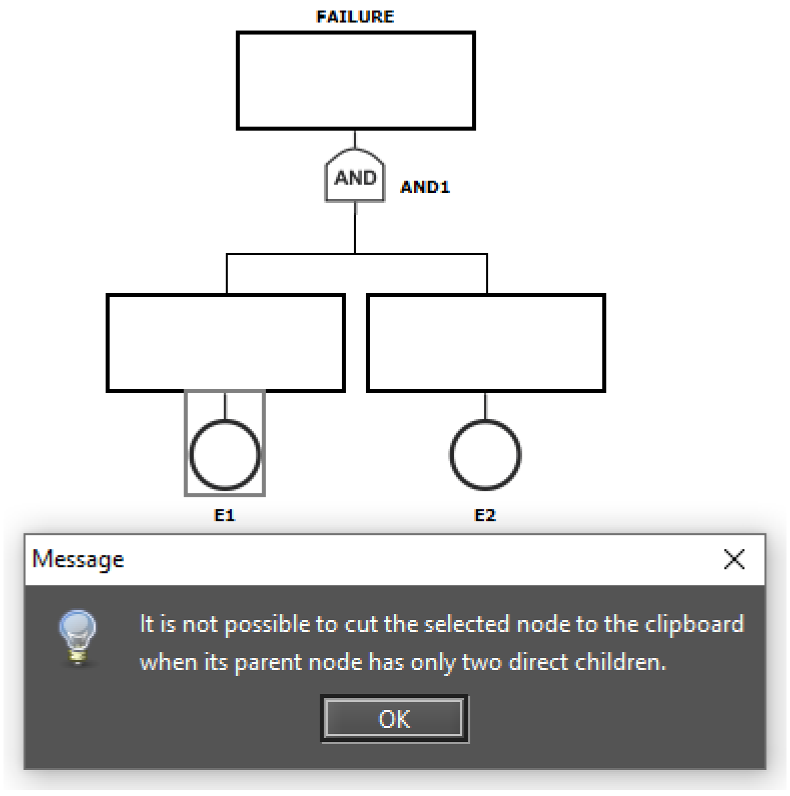


Figure 204: Basic Event Popup Menu

Below, we describe each menu item.

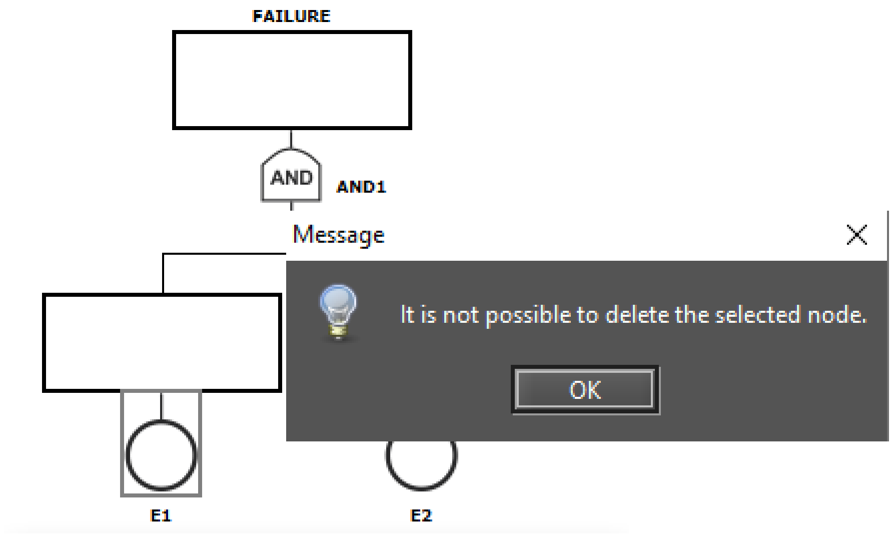
- **Change to a blank event.** Replace the selected node with an empty basic event.
- **Add gate.** Replace the selected basic event with a gate. After choosing the type of gate to be inserted in the submenu, the "Add Gate" dialog box appears (see Figure 193).
- **Copy.** Copy the selected event to the clipboard.
- **Cut.** Cut the selected event to the clipboard. This is only possible when the parent gate of the selected event has at least three direct children. By selecting this option when there are only two direct nodes at the gate, an error will occur, as we can see by looking at Figure 205.



It is not possible to cut the selected node to the clipboard when its parent node has only two direct children.

Figure 205: Error when Cutting a Node

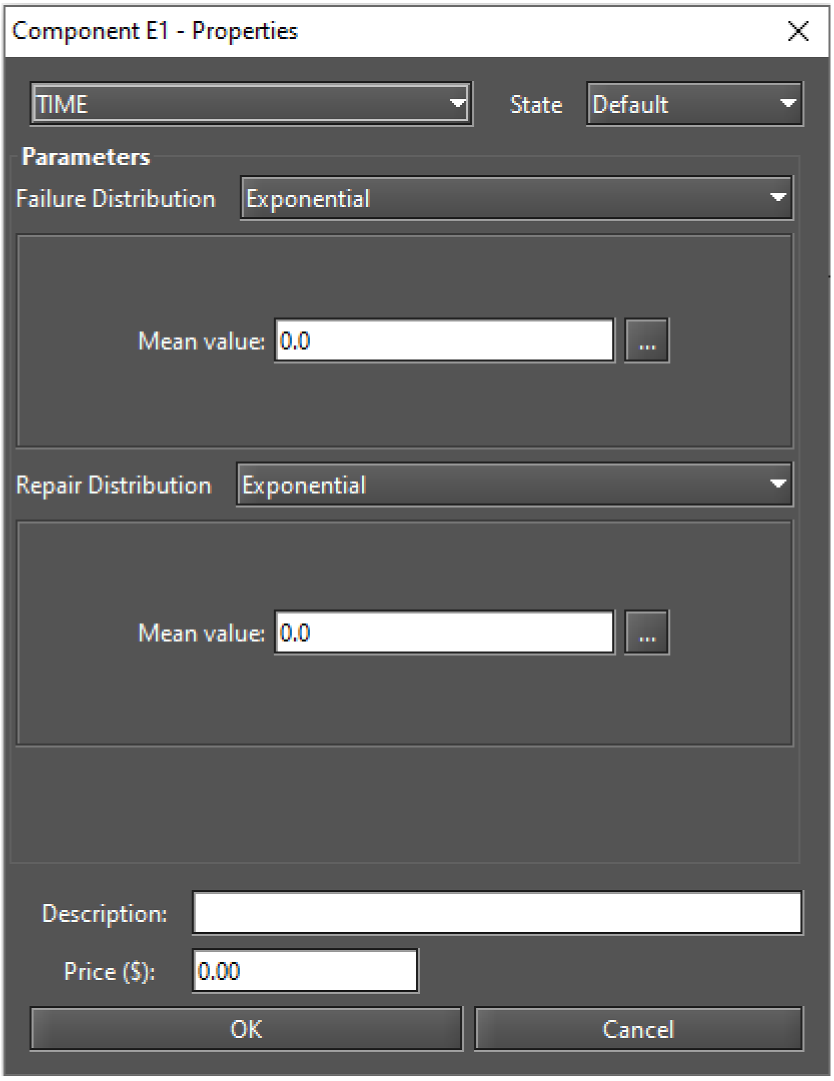
- **Paste.** Replace the select node with the component in the clipboard. It is only enabled when a component has been copied to the clipboard.
- **Delete.** Remove the selected node. It is important to highlight that each gate needs at least two direct nodes. So, when having only two nodes and trying to delete one of them, an error occurs, as we can see by looking at Figure 206.



It is not possible to delete the selected node.

Figure 206: Error when Deleting a Basic Event

- **Properties.** Show the dialog box for changing the properties of the basic event (see Figure 207).



Component E1 - Properties

TIME | State: Default

Parameters

Failure Distribution: Exponential

Mean value: 0.0

Repair Distribution: Exponential

Mean value: 0.0

Description:

Price ($): 0.00

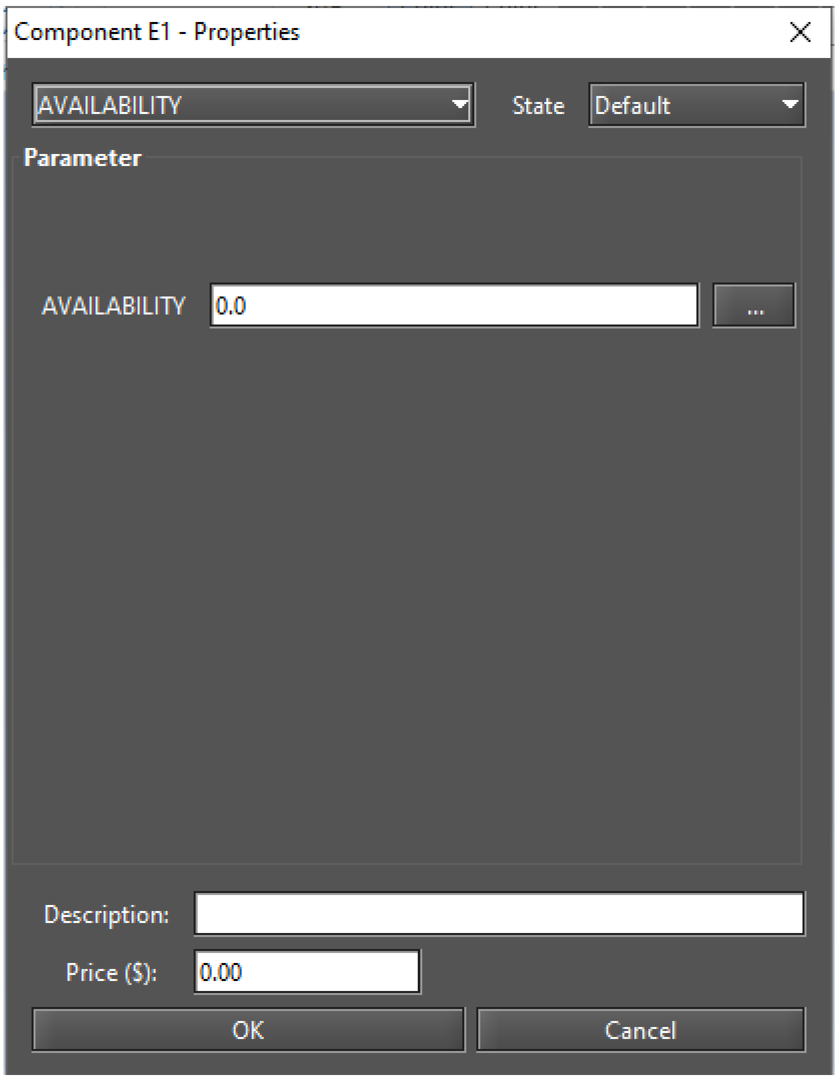
Figure 207: Basic Event Properties Dialog

Below, we describe the fields that appear in the dialog box for updating the properties of a basic event.

- **Parameters Type.** There are four parameters types available to be chosen: RATES, TIME, AVAILABILITY, and RELIABILITY. The default parameter type is TIME.

  When the parameter type is TIME or RATES, the user can enter the values for the failure and repair parameters (see Figure 207). On the other hand, when the type is AVAILABILITY or RELIABILITY, the user must enter the value corresponding to the selected type, as we can see in Figure 208. In this example, the user must enter the availability of the component represented by the basic event.

- **State.** The state of the basic event. There are two states available to be chosen: DEFAULT or FAILED. The default state is DEFAULT. Thus, when having the state parameter as DEFAULT, it means that the failure event has not occurred. On the other hand, having the state parameter as FAILED, it indicates that the failure event has occurred.


Component E1 - Properties

AVAILABILITY — State: Default

Parameter

AVAILABILITY: 0.0

Description:

Price ($): 0.00

Figure 208: Basic Event Properties Dialog - Availability Parameter

- **Failure Parameters.** The Mercury tool supports a large number of probability distributions. Moreover, depending on the chosen distribution, the fields related to the parameters of that distribution appear in order for the user to enter their values. Besides that, the user may attach a label to the failure parameter or define this parameter by means of metrics computed from a CTMC model, which we call as hierarchical composition. See Section 7.1 for further information about hierarchical evaluations.

  The button **"..."** enables the user to select any label declared or metric defined in a CTMC model as hierarchical parameters from submodels.

- **Repair Parameters.** The fields related to the parameters of the chosen probability distribution appear for the user to enter their values. Besides that, the user may attach a label to the repair parameter or define this parameter by means of metrics computed from a CTMC model.

- **Price.** It refers to the cost regarding the component/event represented by the node. That cost is considered when evaluating the component importance and total cost of acquisition. For further information about this evaluation see Section 7.2.4.

Now, let us see the types of basic events and how they are graphically represented. Figure 209 shows the three types of basic events. Figures 209.a and 209.b demonstrate basic events with failure and repair or reliability or availability parameters assigned by the user, but in a) the state parameter is defined as “Default” while in b) it is defined as “Failed”. On the other hand, Figures 209.c shows a basic event with no failure and repair or reliability or availability parameters assigned to it.

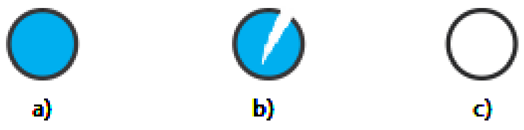


Figure 209: Types of Basic Events

When the required parameters of all basic events linked to a gate are entered, the color of that gate changes to yellow, as we can see by looking at Figure 210.

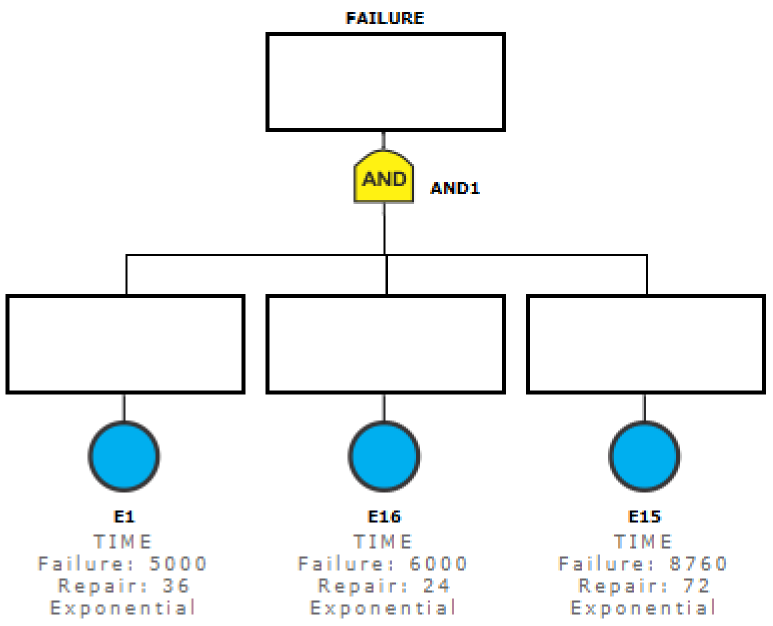


Figure 210: FT Model

7.1 Hierarchical Evaluation

Basic events parameters can be set by evaluation of submodels such as CTMC. This kind of evaluation uses a metric result from a submodel as input to one or more basic event parameters. So, combinatorial and state-space models may be hierarchically combined in order to get the best of both models, e.g., an FT model could be used to represent the dependability relationship between independent subsystems, while more complex fail and repair mechanisms are modeled with CTMCs [4].

Mercury supports evaluation from a CTMC as a submodel to an FT model. At evaluation time, the CTMC will be solved and the chosen parameters will be gotten from that. Thus, it is necessary to have both CTMC and FT models drawn inside of the same Mercury project. For example, let us assume that the following CMTC and FT models exist in a project, as depicted in Figures 211 and 212, respectively.

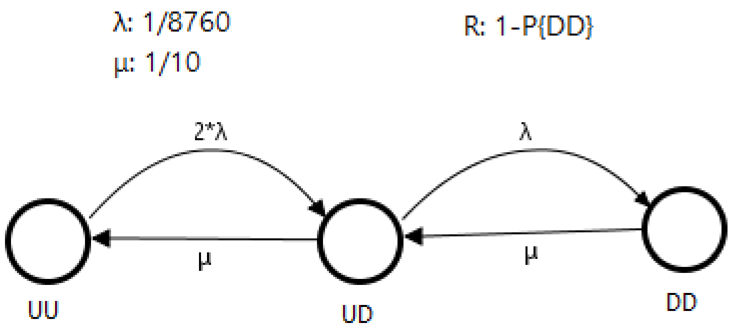


Figure 211: CTMC Submodel

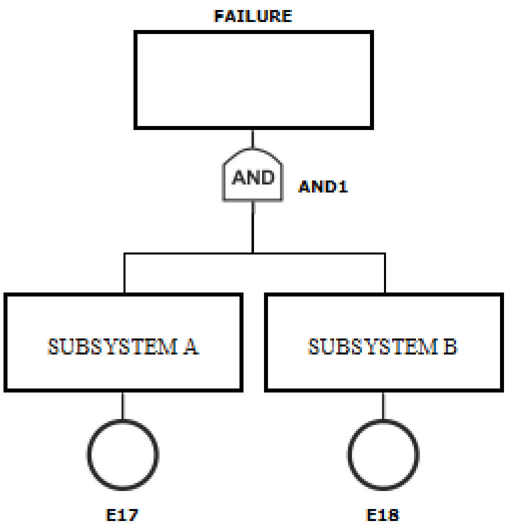
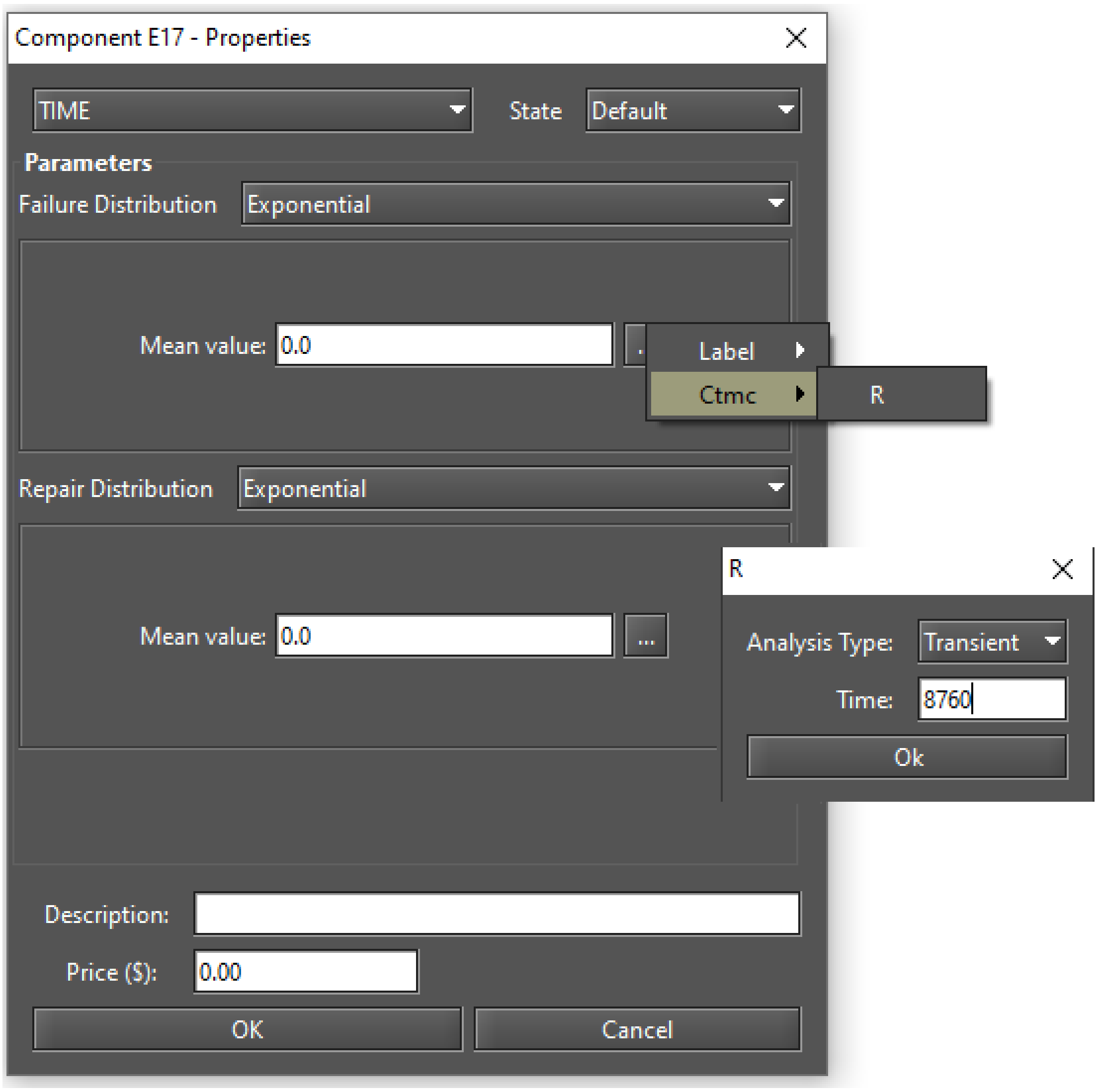


Figure 212: FT Model

When the submodel is created, it is possible to get the parameter from it by clicking on the "..." button adjacent to the parameter, as we can see by looking at the Figure 213. Following, one can choose the desired metric from the submodel on the "Ctmc" submenu and then select the kind of analysis (stationary or transient). When the transient analysis is chosen, it is necessary to provide the time parameter.

In this example, we have shown the hierarchical evaluation for the reliability parameter, but it is also possible to get two parameters simultaneously from the submodel. That is the case when the parameters for MTTF (or failure rate) and MTTR (or repair rate) are exponentially distributed.


Component E17 - Properties

TIME

State: Default

Parameters

Failure Distribution: Exponential

Mean value: 0.0

Repair Distribution: Exponential

Mean value: 0.0

Description:

Price ($): 0.00

R

Analysis Type: Transient

Time: 8760

Figure 213: Hierarchical Evaluation for the Reliability Parameter

7.2 FT Evaluation

The Mercury tool makes it possible to carry out a large number of evaluations on FTs. The tool provides seven functionalities for FT evaluations: Exact Evaluation, Bounds Evaluation, Importance Measures, Experiment, Get Functions, Sensitivity Analysis, and Export to RBD model. All these options are available in the menu Evaluate -> FT Evaluation as we can see in Figure 214. In the next subsections, we will introduce each one of these functionalities.

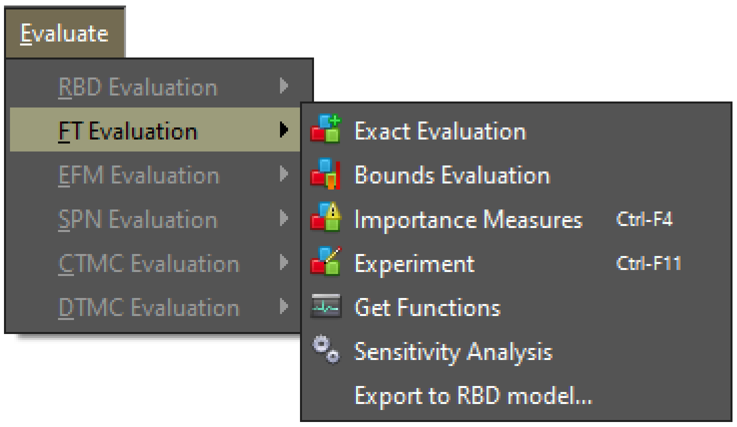
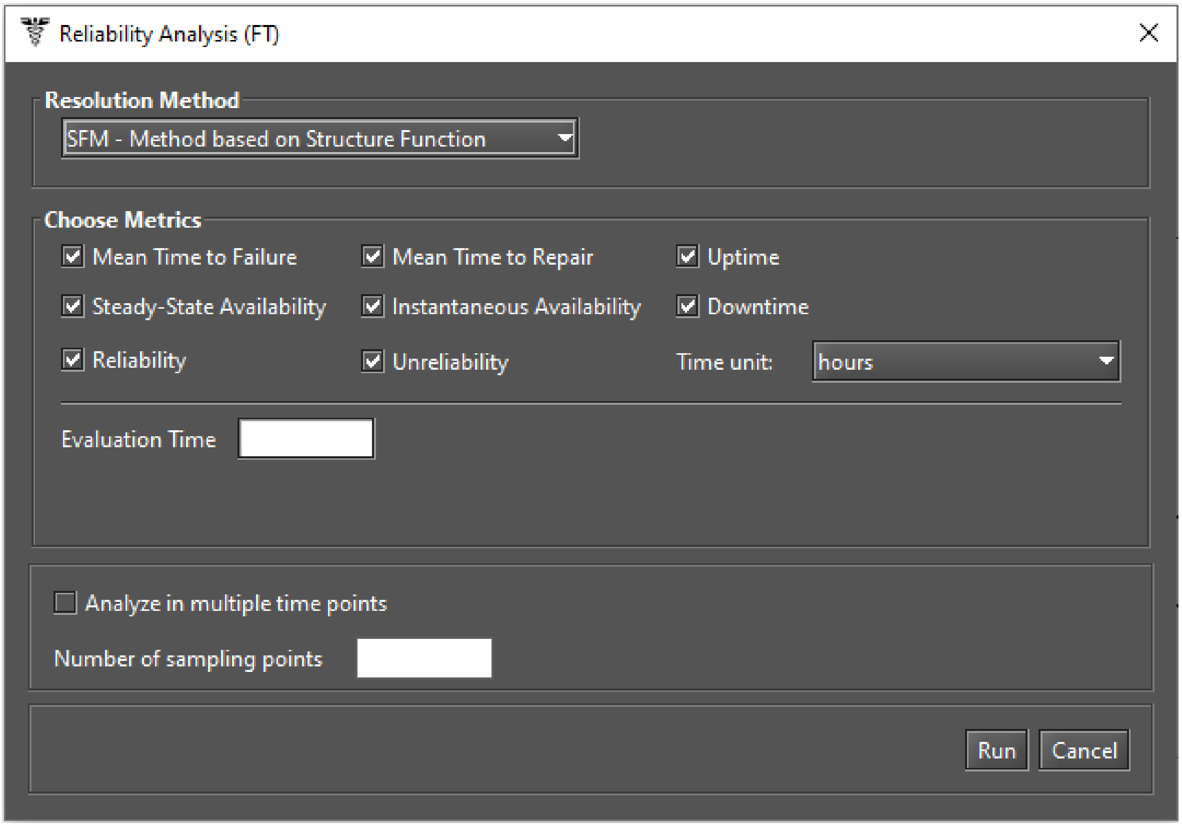


Figure 214: FT Evaluation Menu

7.2.1 Exact Evaluation

The Exact Evaluation makes it possible to carry out a large number of dependability analyses on FT models. This functionality is accessible in the menu Evaluate -> FT Evaluation -> Exact Evaluation. Figure 215 shows the dialog box for dependability evaluation on FTs.

As we can see, users can evaluate eight metrics. The metrics they can choose are Mean Time to Failure, Mean Time to Repair, Steady-State Availability, Instantaneous Availability, Reliability, Unreliability, Uptime, and Downtime. Also, it is possible to choose the time unit to be considered when computing Uptime and Downtime: seconds, minutes, hours, and days. When time-dependent metrics are chosen — Reliability, Unreliability, or Instantaneous Availability —-, users need to enter the time parameter — "Evaluation Time" field. Besides that, there is an option that analyze the selected time-dependent metrics by considering multiple points on time. The time interval from 0 to the evaluation time is divided by considering the number of points entered and, after that, the metric is computed for each point.


Reliability Analysis (FT)

Resolution Method

SFM - Method based on Structure Function

Choose Metrics

- [✓] Mean Time to Failure
- [✓] Mean Time to Repair
- [✓] Uptime
- [✓] Steady-State Availability
- [✓] Instantaneous Availability
- [✓] Downtime
- [✓] Reliability
- [✓] Unreliability

Time unit: hours

Evaluation Time

- [ ] Analyze in multiple time points

Number of sampling points

Figure 215: Input window for FT Analysis

The Mercury tool provides two methods for computing dependability metrics. The user must choose between the SFM (Method based on Structure Function) or SDP (Sum of Disjoint Products) methods, as depicted in the Figure 216. SFM computes metrics by considering the structural function of the FT model. On the other hand, the SDP method, based on Boolean algebra, computes metrics by considering the minimal cuts and minimum paths.

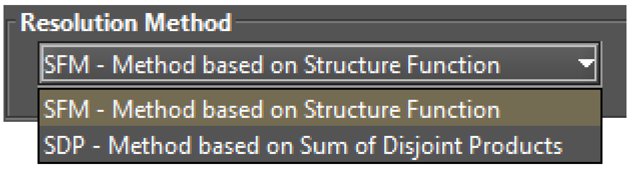

Resolution Method
SFM - Method based on Structure Function
SFM - Method based on Structure Function
SDP - Method based on Sum of Disjoint Products

Figure 216: Resolution Method

After choosing the desired options, and entered the evaluation time and number of sampling points, if required, the user must click on the “Run” button in order to start the evaluation.

Now, let us demonstrate how to perform evaluations by using the Mercury tool. We have considered an FT model having four events, each one contributing to the system's overall failure, as depicted by Figure 217. As we can see, the system fails when the events **e1** or **e8** or both events in the AND gate — **e3** and **e4** — occur. Node e8 represents a k-out-of-n component. This means that in order to the event e8 occurs it is necessary that at least 3 of the 5 components fail.

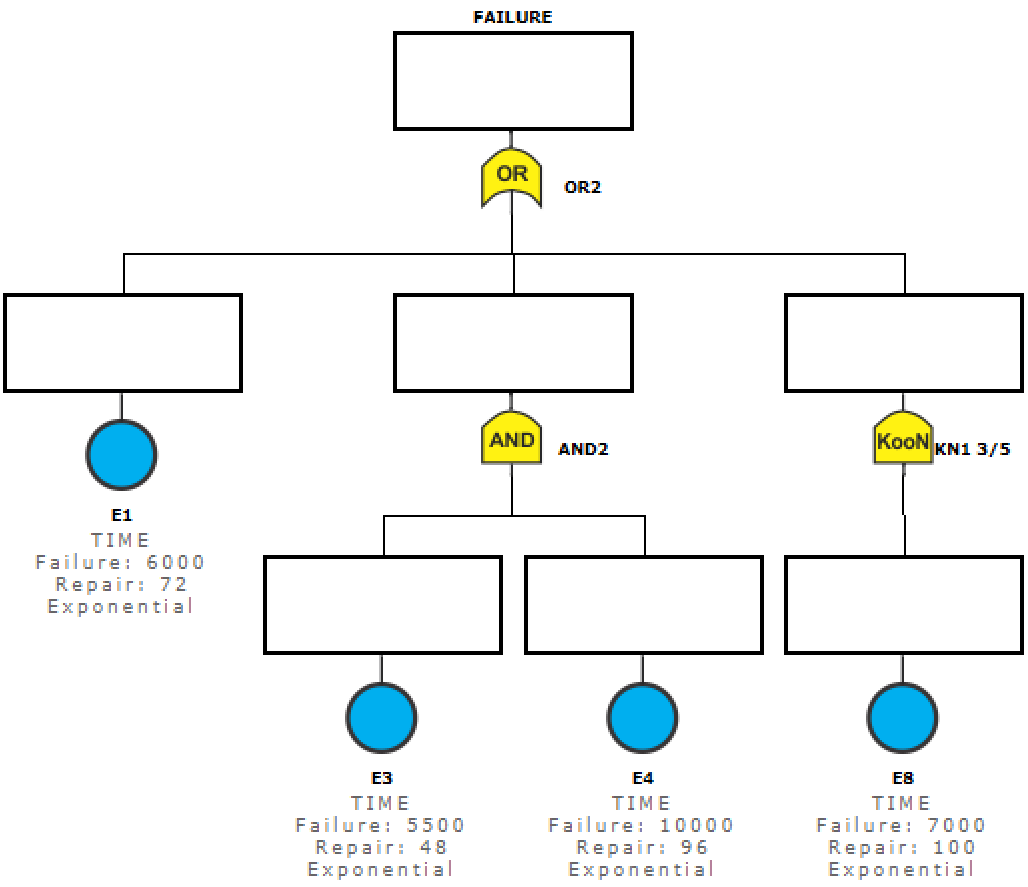
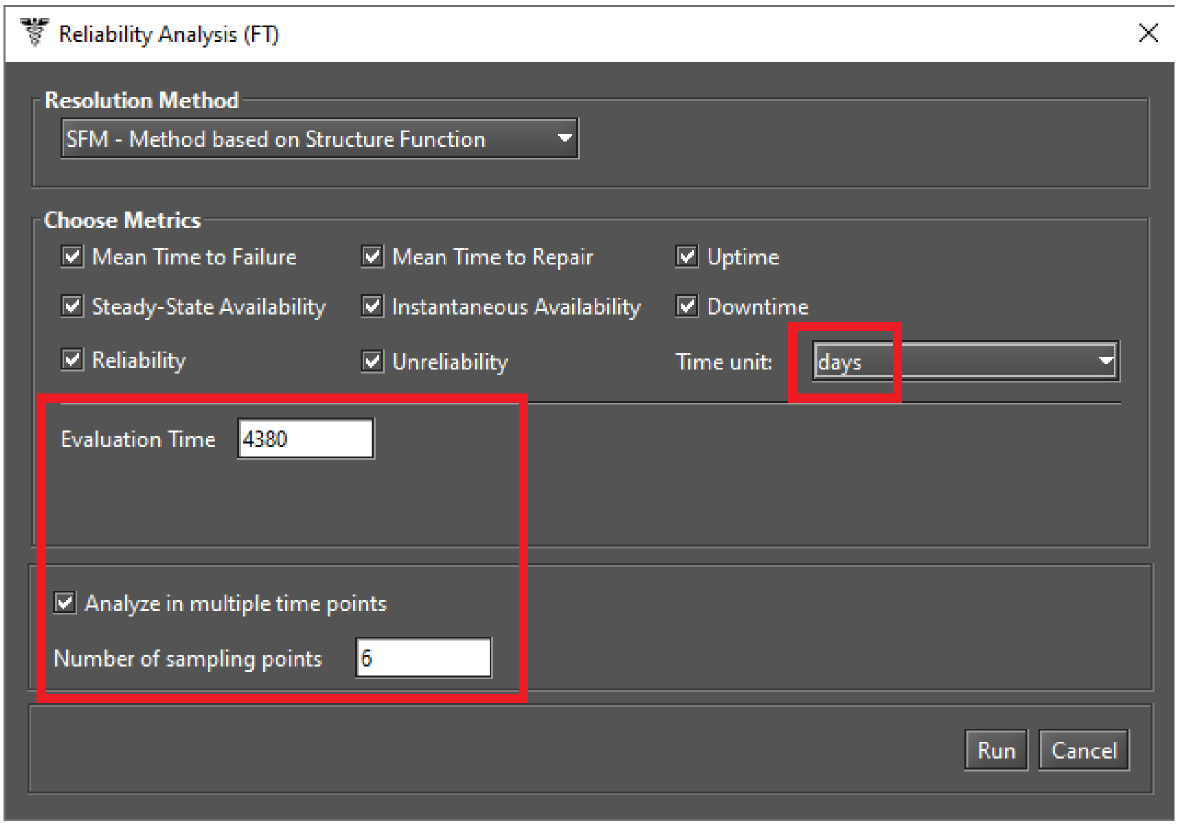


Figure 217: FT Model for Dependability Evaluation

We carried out an evaluation by considering the time parameter equal to 4380h, "days" as time unit, and six sampling points (see Figure 218). At the end of the evaluation, a dialog box appears showing the results, as we can see in Figure 219. In this window, the results are presented in two groups. There exist the "Steady-state Results" for steady-state metrics and "Instantaneous Results" for time-dependent metrics. Listing 8 shows the results in detail.


Reliability Analysis (FT)

Resolution Method: SFM - Method based on Structure Function

Choose Metrics
- [✓] Mean Time to Failure
- [✓] Mean Time to Repair
- [✓] Uptime
- [✓] Steady-State Availability
- [✓] Instantaneous Availability
- [✓] Downtime
- [✓] Reliability
- [✓] Unreliability

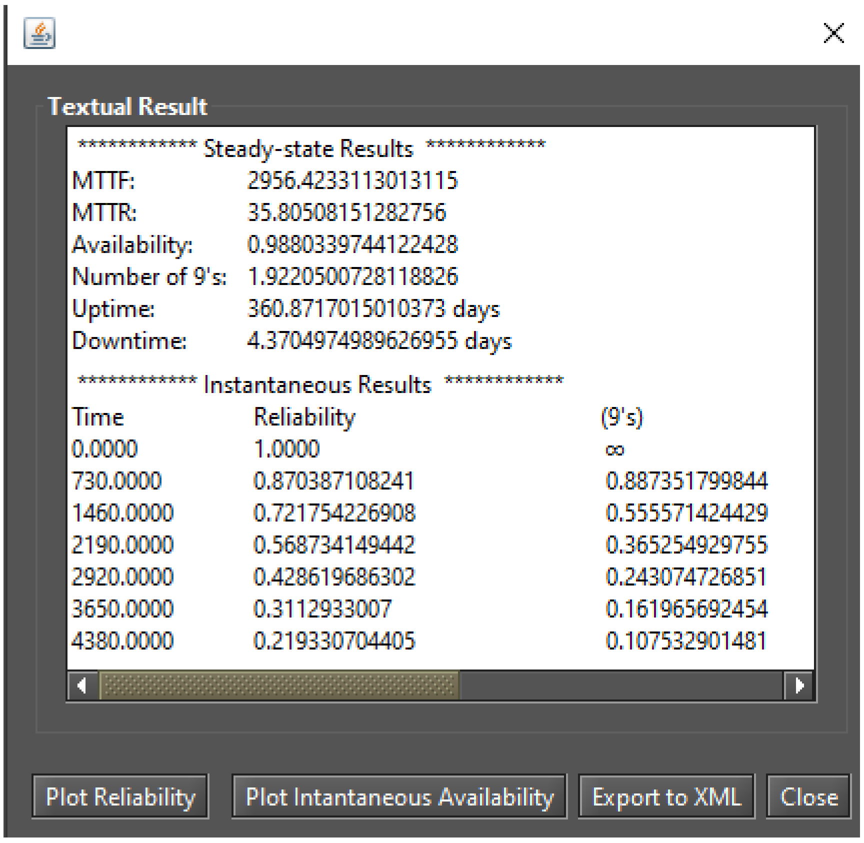
Time unit: days

Evaluation Time: 4380

[✓] Analyze in multiple time points

Number of sampling points: 6

Figure 218: FT Analysis


Textual Result

```
************ Steady-state Results ************
MTTF:           2956.4233113013115
MTTR:           35.80508151282756
Availability:   0.9880339744122428
Number of 9's:  1.9220500728118826
Uptime:         360.8717015010373 days
Downtime:       4.3704974989626955 days

************ Instantaneous Results ************
Time            Reliability            (9's)
0.0000          1.0000                 ∞
730.0000        0.870387108241         0.887351799844
1460.0000       0.721754226908         0.555571424429
2190.0000       0.568734149442         0.365254929755
2920.0000       0.428619686302         0.243074726851
3650.0000       0.3112933007           0.161965692454
4380.0000       0.219330704405         0.107532901481
```

Figure 219: Dependability Results

Listing 8: Dependability Results for the FT Model

```
************ Steady-state Results  ************
MTTF:            2956.4233113013115
MTTR:            35.80508151282756
Availability:    0.9880339744122428
Number of 9's:   1.9220500728118826
Uptime:          360.8717015010373 days
Downtime:        4.3704974989626955 days


************ Instantaneous Results  ************
Time         Reliability     (9's)           Unreliability   Inst. availability
0.0000       1.0000                          0.0000          1.0000
730.0000     0.870387108241  0.887351799844  0.129612891759  0.988034475869
1460.0000    0.721754226908  0.555571424429  0.278245773092  0.988033974474
2190.0000    0.568734149442  0.365254929755  0.431265850558  0.988033974412
2920.0000    0.428619686302  0.243074726851  0.571380313698  0.988033974412
3650.0000    0.311293300700  0.161965692454  0.688706699300  0.988033974412
4380.0000    0.219330704405  0.107532901481  0.780669295595  0.988033974412
```

Figure 220 shows the “Reliability Chart” dialog box displayed when clicking on the “Plot Reliability” button. This window is only displayed when the reliability metric is selected in the input dialog box. The number of points on the plotted lines is determined by the number of points entered by the user.

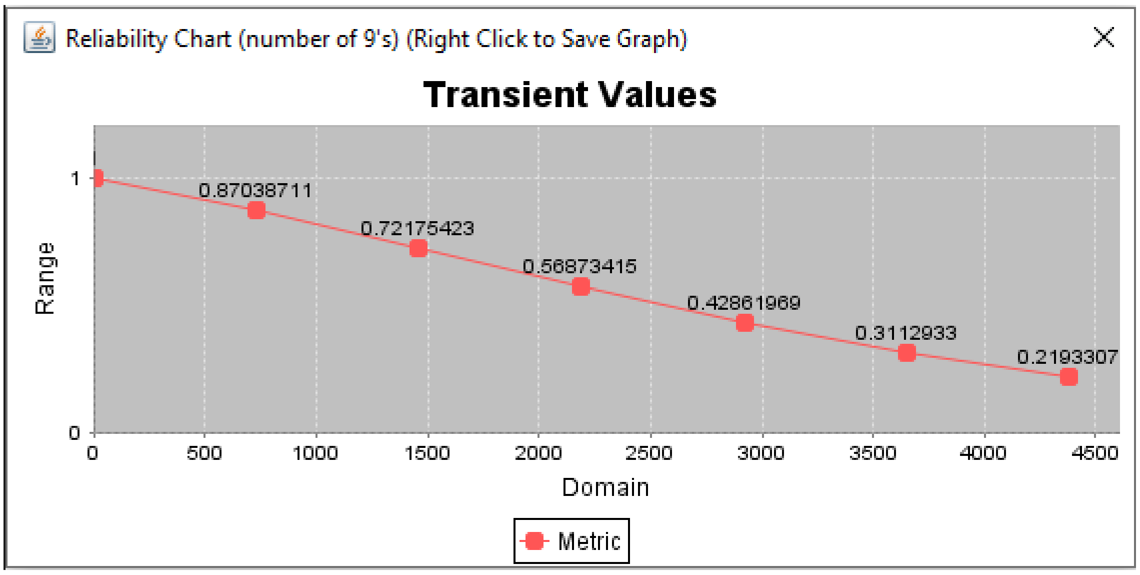


Figure 220: Plotting Reliability

The resolution of an FT through simulation occurs when the probability distribution regarding failure and/or repair of one or more components is non-exponential, such as Erlang, Gamma, among others. Figure 221 depicts a model having a node with non-exponential parameters (node e1). When this occurs, non-exponential nodes are converted into SPNs and then they are solved by simulation. When the Mercury tool detects this, it shows a message informing that hierarchical analysis has been detected, as we can see by looking at Figure 222. Now, the tool makes available two resolution methods to be chosen: *SFM - Simulation* and *SDP - Simulation*.

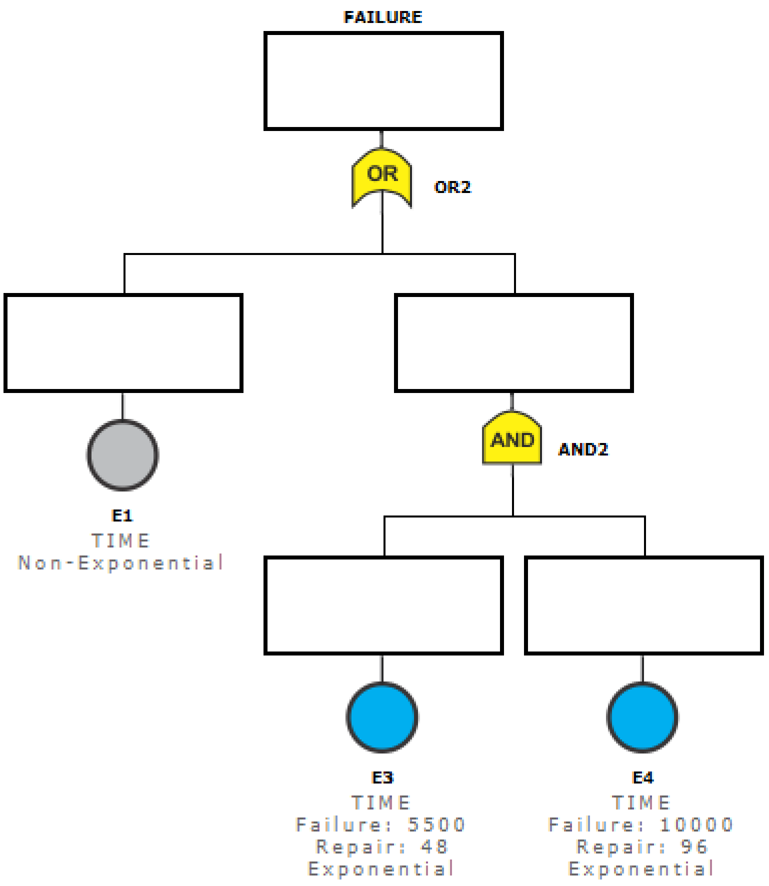
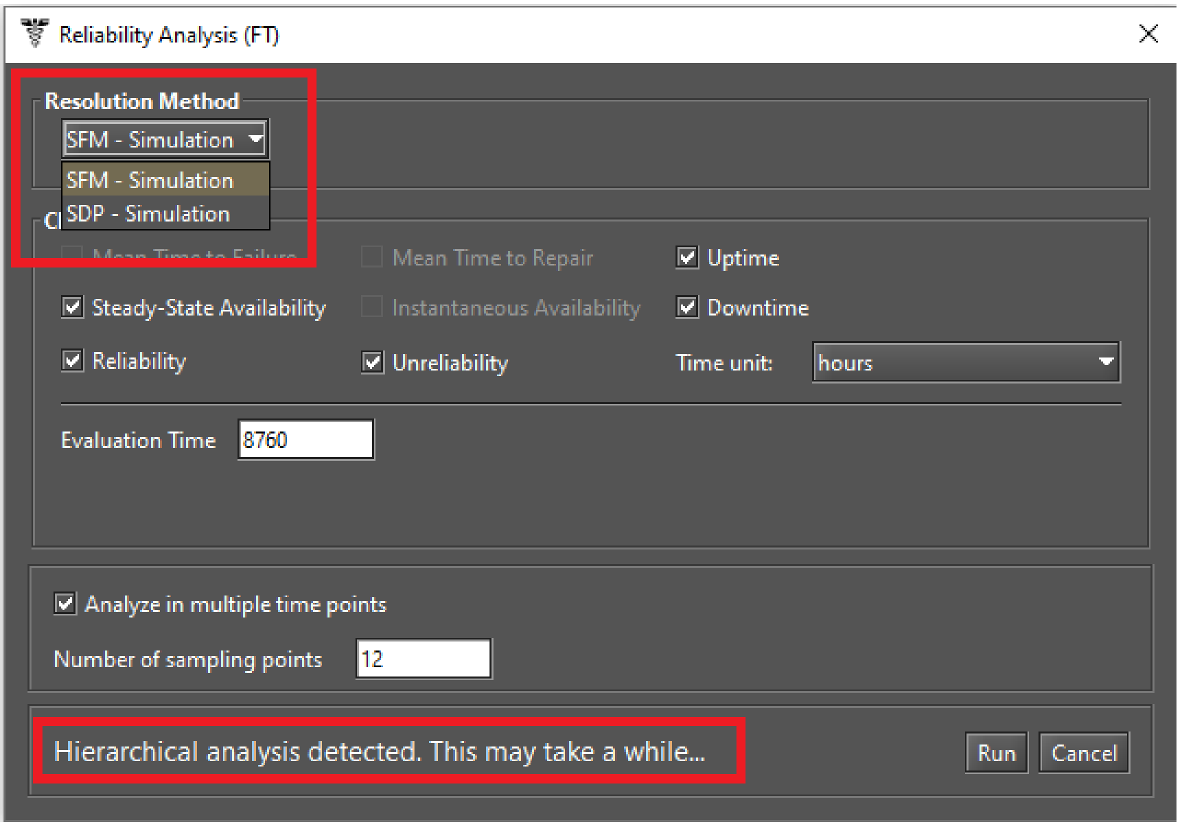


Figure 221: FT Model Having a Non-Exponential Node

When clicking on the “Run” button, the user is prompted to provide some input parameters for simulation, according to the chosen metrics. When only steady-state metrics are selected, the dialog box presented by Figure 223 is shown. On the other hand, when one or more time-dependent metric is selected, the dialog box presented by Figure 224 is shown.


Reliability Analysis (FT)

Resolution Method: SFM - Simulation

SFM - Simulation
SDP - Simulation

[ ] Mean Time to Failure
[ ] Mean Time to Repair
[✓] Uptime
[✓] Steady-State Availability
[ ] Instantaneous Availability
[✓] Downtime
[✓] Reliability
[✓] Unreliability
Time unit: hours

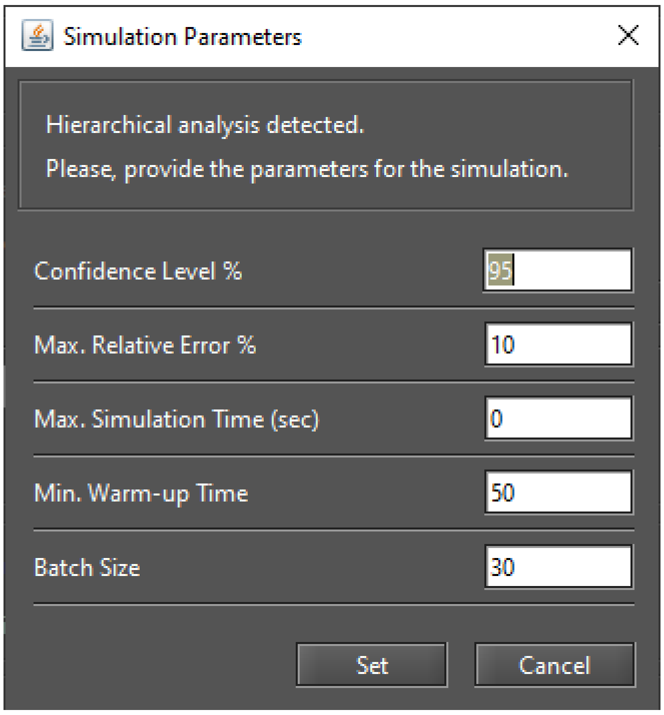
Evaluation Time: 8760

[✓] Analyze in multiple time points

Number of sampling points: 12

Hierarchical analysis detected. This may take a while...

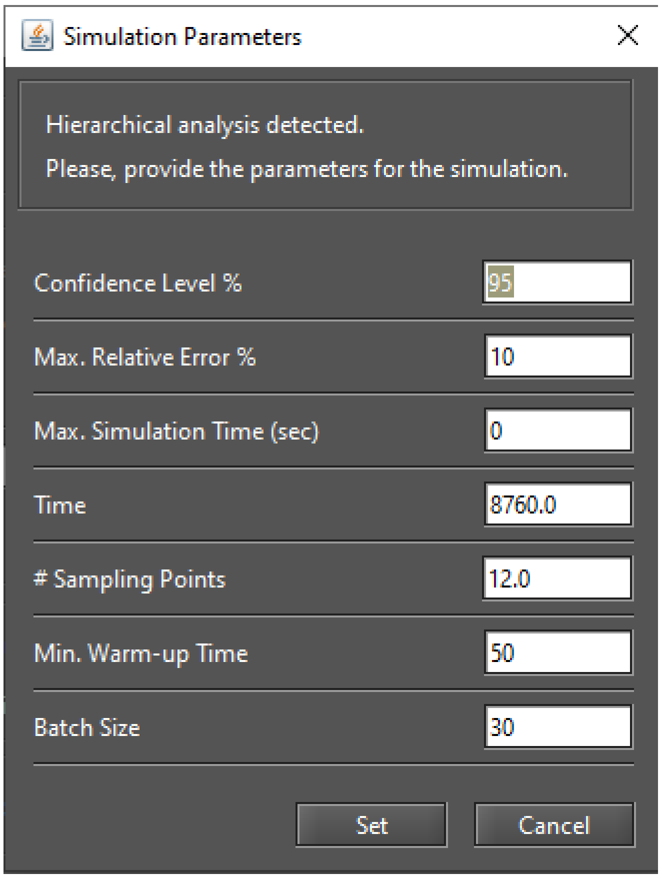
Figure 222: Input window for FT Hierarchical Analysis


Simulation Parameters

Hierarchical analysis detected.
Please, provide the parameters for the simulation.

Confidence Level %	95
Max. Relative Error %	10
Max. Simulation Time (sec)	0
Min. Warm-up Time	50
Batch Size	30

Figure 223: FT Simulation for Steady-State Metrics


Simulation Parameters

Hierarchical analysis detected.
Please, provide the parameters for the simulation.

Confidence Level %	95
Max. Relative Error %	10
Max. Simulation Time (sec)	0
Time	8760.0
# Sampling Points	12.0
Min. Warm-up Time	50
Batch Size	30

Figure 224: FT Simulation for Time-Dependent Metrics

After entering the required parameters and starting the simulation, the results are presented. It is important to highlight that when solving FTs by simulation, it is not possible to solve the following metrics: mean time to failure, mean time to repair, and instantaneous availability. For further information about the input parameters for simulation, see Section 2.1.

7.2.2 FT Experiment

Sometimes, it is necessary to evaluate the impact of varying some parameters on the required metrics. The Mercury tool provides a feature for supporting this type of evaluation on fault trees. Now, we will demonstrate an example of running an experiment on an FT model.

The first step for that aim is to define one or more labels. Labels are variables that store numeric values and they can be attached to failure/repair parameters of basic events. The value of a label is changed by considering the data entered by the user and, in each changing, the selected metric is evaluated.

To insert a label, the user must right-click on the left-side label panel of the FT tab and, when the pop-up menu appears, she must click on the “Insert label” menu item, as depicted by Figure 225.

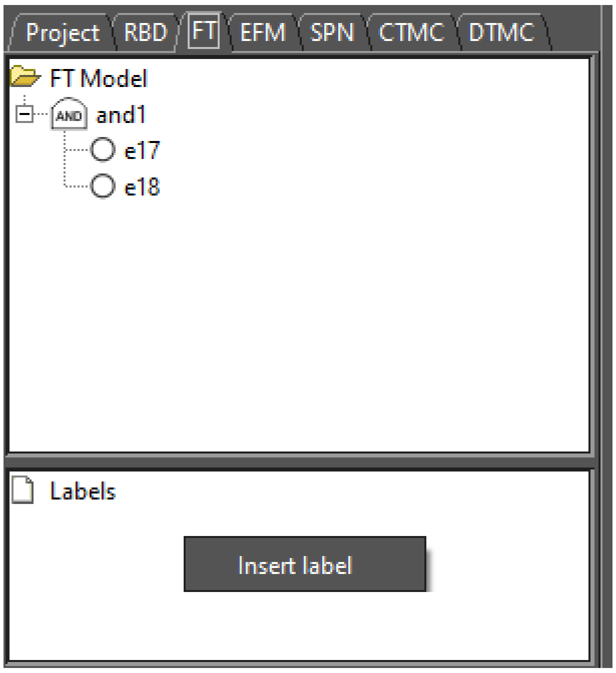
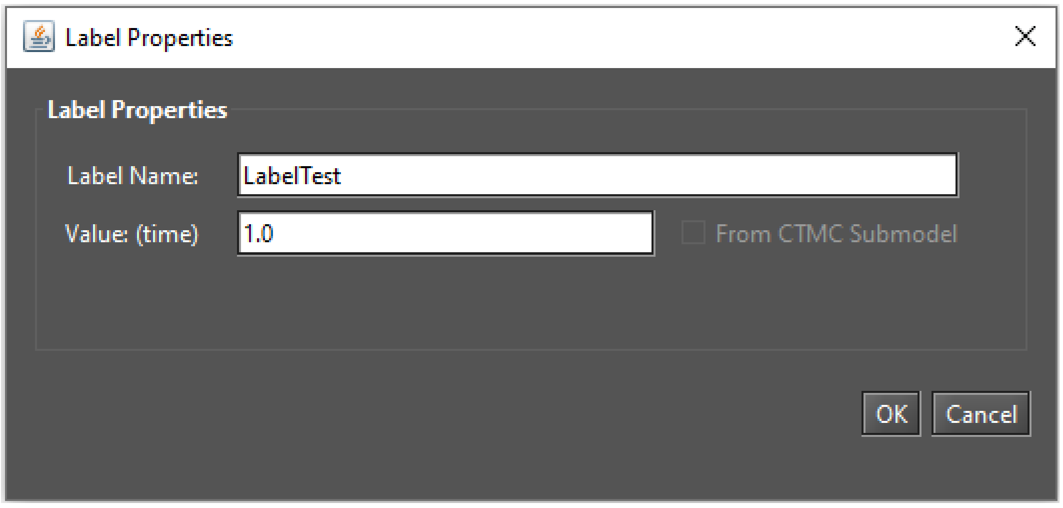


Figure 225: Inserting a Label for the FT Model

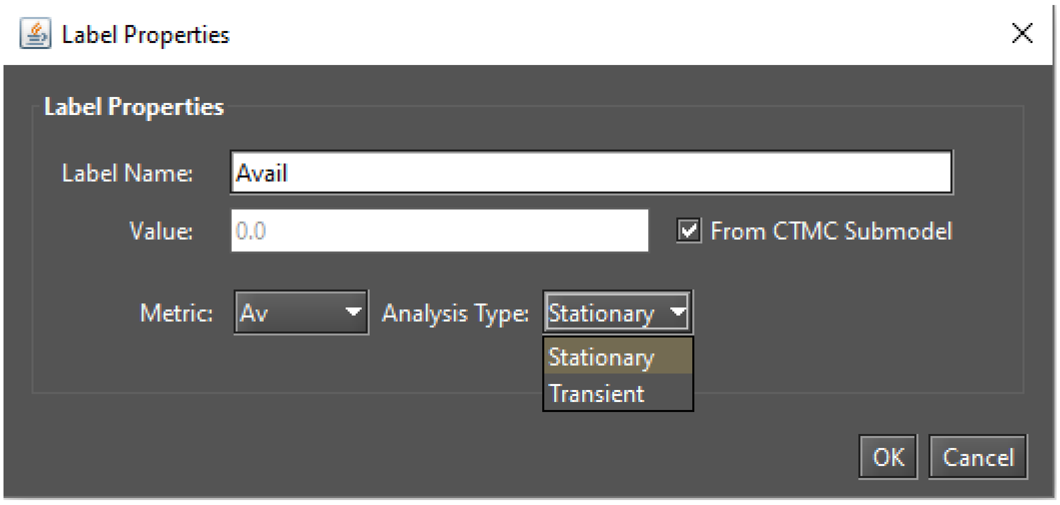
After that, the “Label Properties” input dialog box is shown (see Figure 226). In this dialog box, the user must set the properties of the label.



Label Name: LabelTest
Value: (time) 1.0

Figure 226: FT Label Properties

The user can define the failure/repair parameter of basic events by means of metrics computed from a CTMC model, which we call as hierarchical composition. This is done by assigning a CTMC metric to a label, as shown in Figure 227. The user must check the box "From CTMC submodel", and select one of the metrics previously defined in the CTMC. The type of analysis (Stationary or Transient) must be selected too. In the case of transient analysis, the time parameter is also required.


Label Properties

Label Properties

Label Name: Avail

Value: 0.0 [✓ From CTMC Submodel]

Metric: Av Analysis Type: Stationary

Figure 227: Assigning a CTMC Metric to an FT Label

When a label is inserted, it is available in the left-side panel of the FT tab (see Figure 228).

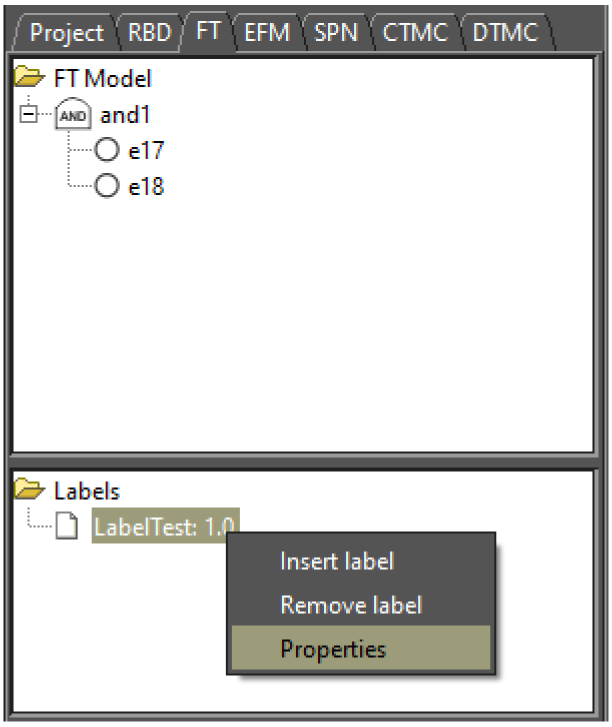
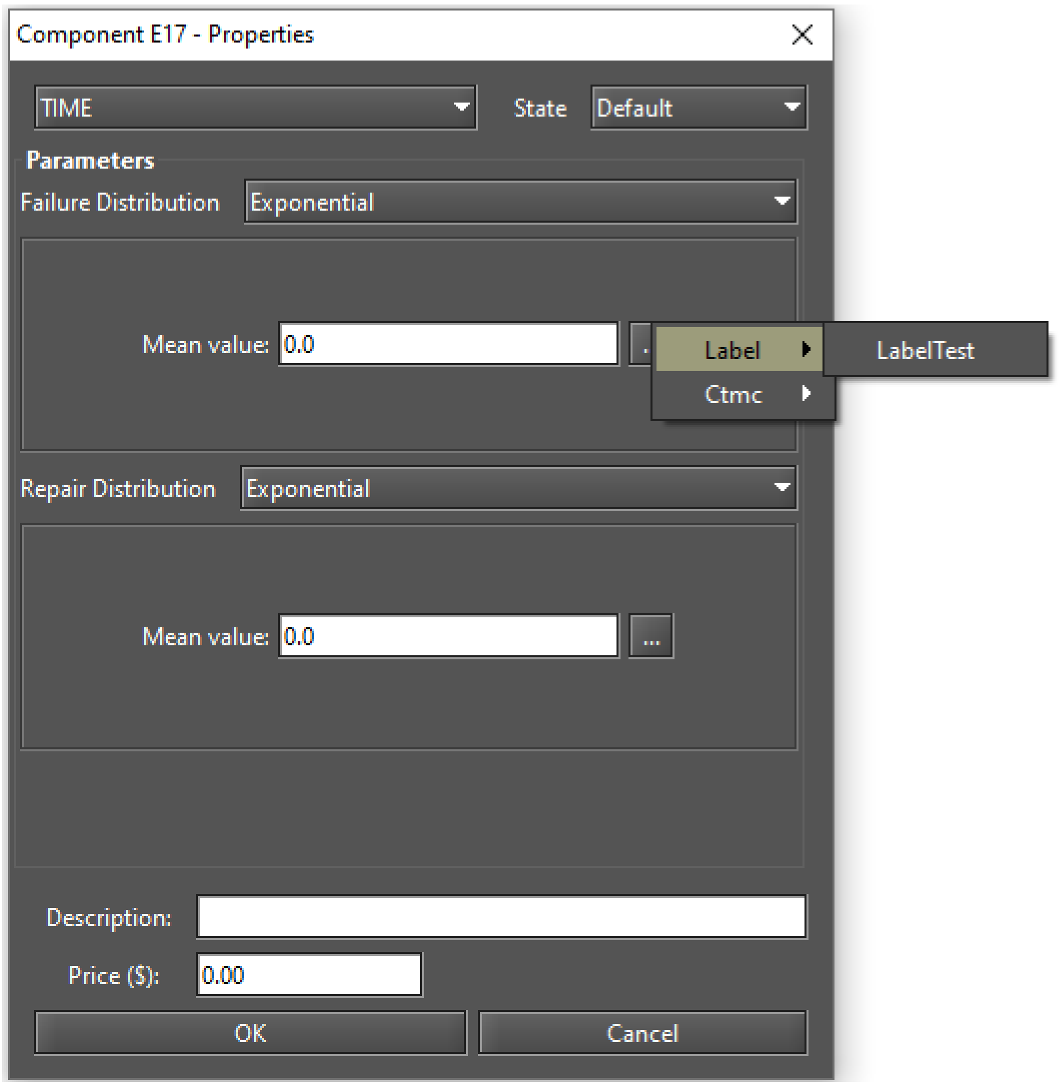


Figure 228: FT Left-Side Panel

By right-clicking on any inserted label, a pop-up menu with three menu items appears, as we can see by looking at the last figure. Below, we describe each of them.

- **Insert label.** Show the “Label Properties” dialog box that allows inserting a new label.
- **Remove label.** Remove the selected label.
- **Properties.** Show the “Label Properties” dialog box that allows changing the properties of the label.

After defining a label, it is necessary to attach this label to the failure/repair parameter of the basic event under evaluation. Figure 229 demonstrates how to attach a label to a failure/repair parameter of a basic event.


Component E17 - Properties

TIME

State: Default

Parameters

Failure Distribution: Exponential

Mean value: 0.0

Label ▸ LabelTest

Ctmc ▸

Repair Distribution: Exponential

Mean value: 0.0

Description:

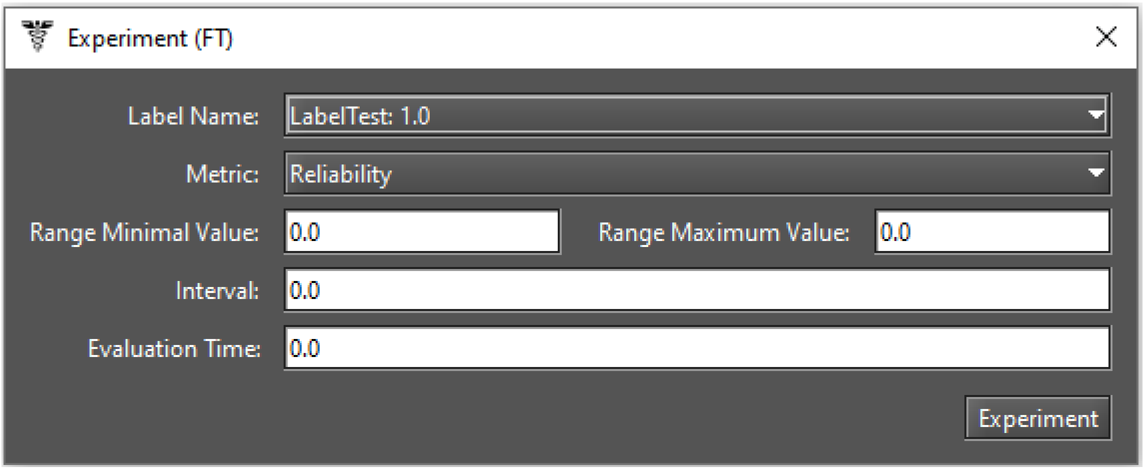
Price ($): 0.00

Figure 229: Attaching a Label to a Basic Event Parameter

The feature for FT experiment is accessible in the menu Evaluate -> FT Evaluation -> Experiment. Figure 230 shows the “FT Experiment” dialog. In order to perform the experiment, the user must fill the required fields. Below, we describe each of them.

- **Label Name.** The label to have its value changed.
- **Metric.** The metric to be evaluated.
- **Minimum Value.** The initial value to be assigned to the selected label.
- **Maximum Value.** The final value to be assigned to the selected label.
- **Interval.** It is the step-size for changing the value of the label. The label starts with the minimum value and its value is increased by considering the entered interval. At each change, the selected metric is evaluated. The experiment ends when the maximum value for the label is reached.

- **Evaluation Time.** The evaluation time to be considered. For time-dependent metrics — reliability, unreliability, instantaneous availability — the user must enter the time parameter.


Experiment (FT)

Label Name:	LabelTest: 1.0		
Metric:	Reliability		
Range Minimal Value:	0.0	Range Maximum Value:	0.0
Interval:	0.0		
Evaluation Time:	0.0		

Figure 230: FT Experiment Dialog

After defining the input parameters for the experiment, the user must click on the "Experiment" button in order to start the experiment. After that, when the experiment has been finished, the "Experiment Result" dialog is shown as depicted by Figure 231.

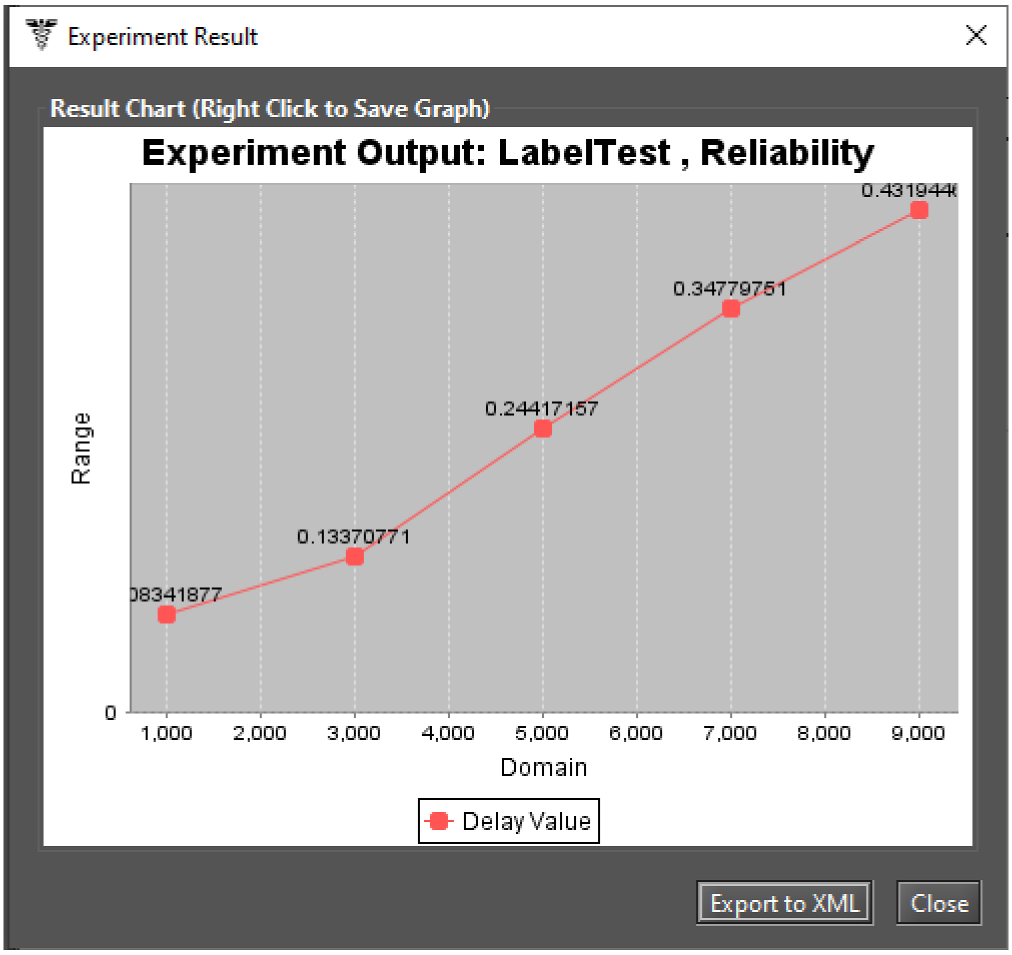


Figure 231: Experiment Result Dialog

7.2.3 Bounds for Dependability Analisys

Bounds for Dependability Analysis is adopted to estimate the dependability metrics through the calculation of reliability, availability, or downtime. Therefore, it is necessary to estimate the bounds values (upper and lower limits) for computing this analysis in which users quickly obtain the results. This functionality should be performed when the model in analysis is very large. This method is subdivided into two parts: (i) computing the bounds values and (ii) adopting the sum of disjoint points to find the successive values and the number of iterations.

It is accessible in the menu Evaluate -> FT Evaluation -> Bounds Evaluation. Figure 232 shows the “Bounds for Dependability Analysis” dialog box. As we can see in the Figure 233, the user can evaluate four metrics: Steady State Availability, Instantaneous Availability, Reliability, and Downtime. The user needs to enter the time parameter for time-dependent metrics. After selecting the desired metric and entering the time, if required, the user needs to click on the button “Get Start Values” in order to start the evaluation.

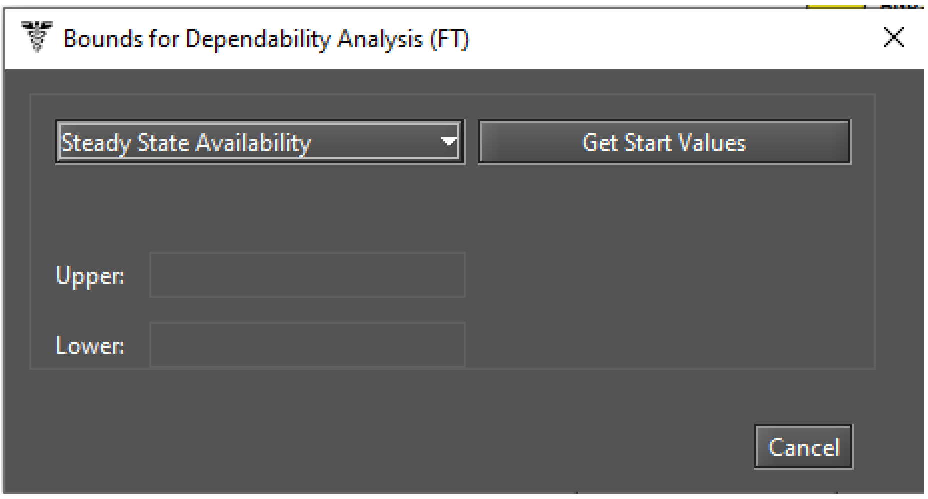

Bounds for Dependability Analysis (FT)
Steady State Availability
Get Start Values
Upper:
Lower:
Cancel

Figure 232: Bounds for Dependability Analysis

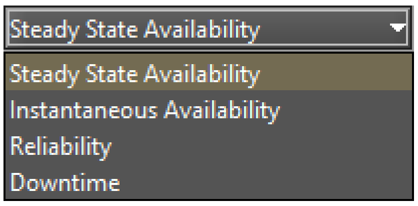

Steady State Availability
Steady State Availability
Instantaneous Availability
Reliability
Downtime

Figure 233: Metrics for Bounds Evaluation

Now, let us demonstrate how to perform bounds evaluation by using the Mercury tool. We have considered an FT model having five events, each one contributing to the system’s overall failure, as depicted in Figure 234. As we can see, the system fails when the events **e1**, **e6** and at least one of the following events occur: **e3**, **e4**, or **e5**. We evaluated the bounds of the reliability for this model by considering the time parameter equal to 8760h (see Figure 235).

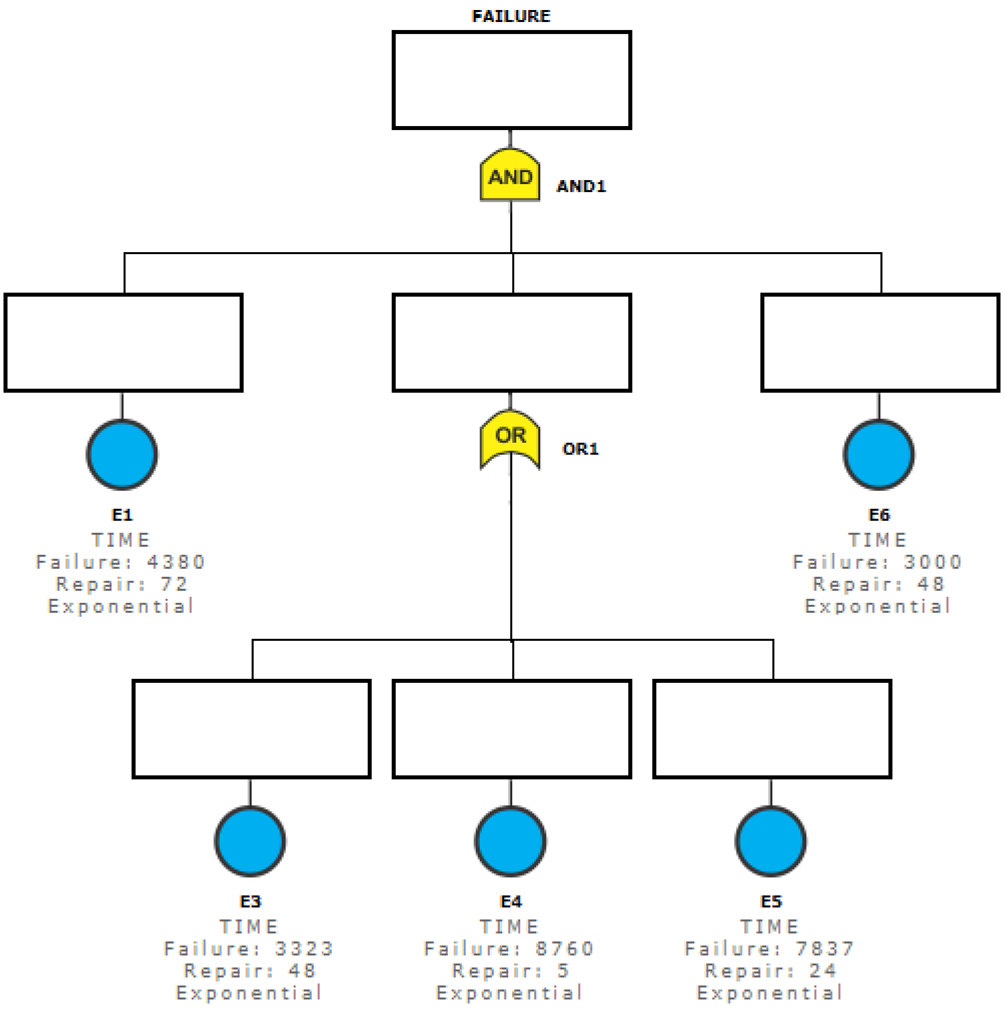
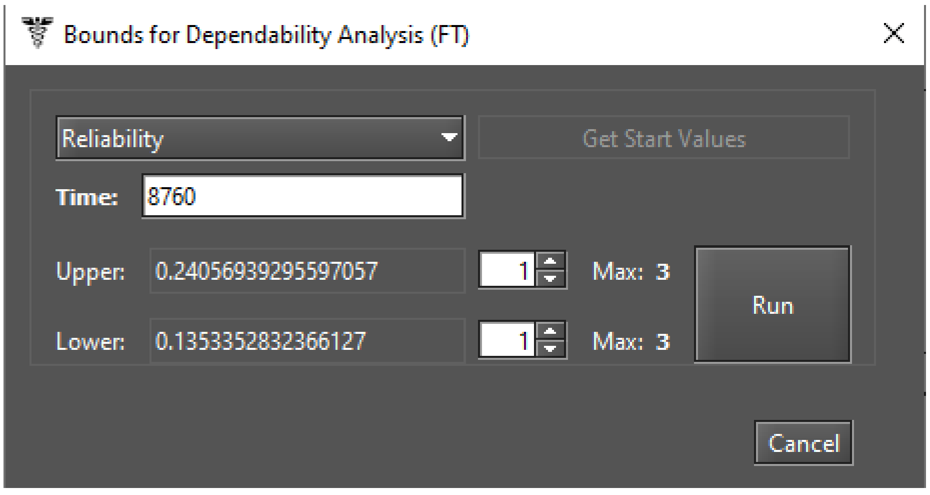


Figure 234: FT Model for Bounds Evaluation


Bounds for Dependability Analysis (FT)

Reliability

Get Start Values

Time: 8760

Upper: 0.24056939295597057 1 Max: 3

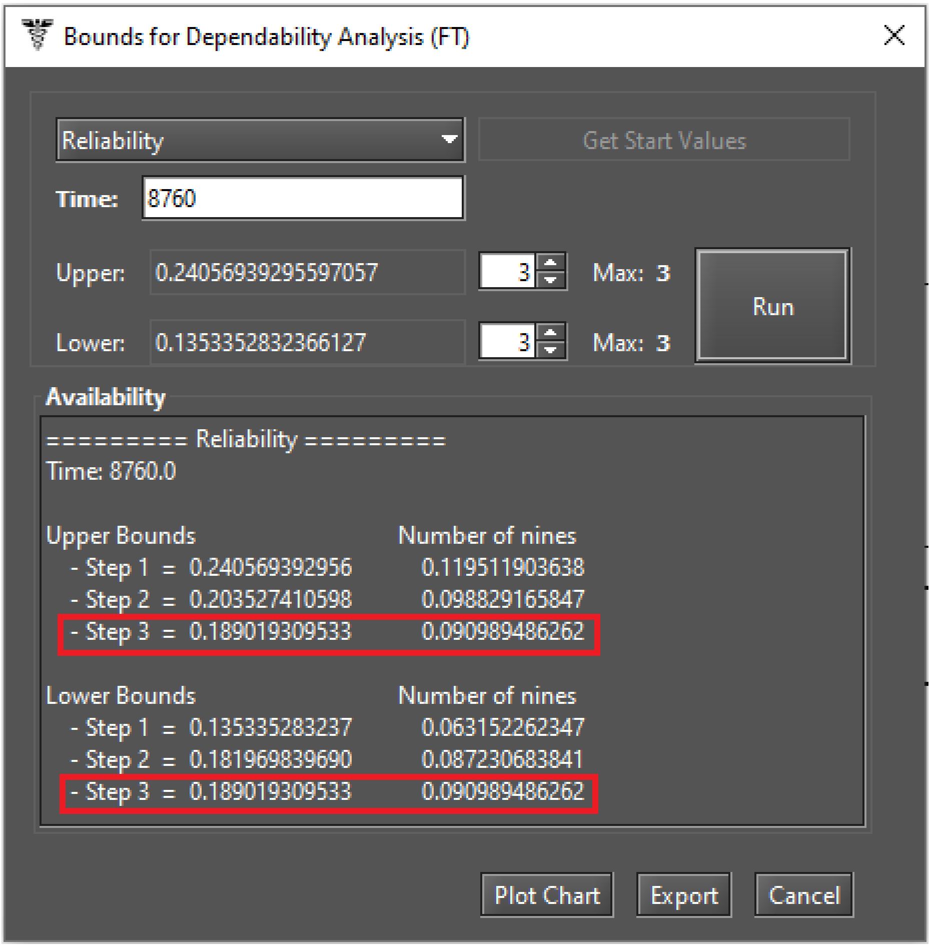
Lower: 0.1353352832366127 1 Max: 3

Run

Cancel

Figure 235: Bounds for Reliability Metric

First, we have computed the upper and lower values. This option adopts the first path and the first cut to provide the higher and lower values of the chosen metric. The paths are related to the lower bounds, in which a minimal set of components is chosen to guarantee the operational mode of the system. Meanwhile, the cuts are related to the upper bounds, in which a minimal set of components that ensure the system on the failure mode is adopted. After getting the upper and lower values, we defined the number of steps to three for the upper and lower values and we ran the evaluation. Now, we can see the result demonstrated by Figure 236. We have highlighted the values for the upper and lower bounds of the last step — step 3. As we can see, the values are equal. By clicking on the "Plot Chart" button we can see the lower and upper bounds of the three steps converging (see Figure 237).



Reliability
Time: 8760
Upper: 0.24056939295597057 3 Max: 3
Lower: 0.1353352832366127 3 Max: 3

Availability

```
========= Reliability =========
Time: 8760.0

Upper Bounds                   Number of nines
  - Step 1  =  0.240569392956    0.119511903638
  - Step 2  =  0.203527410598    0.098829165847
  - Step 3  =  0.189019309533    0.090989486262

Lower Bounds                   Number of nines
  - Step 1  =  0.135335283237    0.063152262347
  - Step 2  =  0.181969839690    0.087230683841
  - Step 3  =  0.189019309533    0.090989486262
```

Figure 236: Bounds for Reliability - Result of Computation

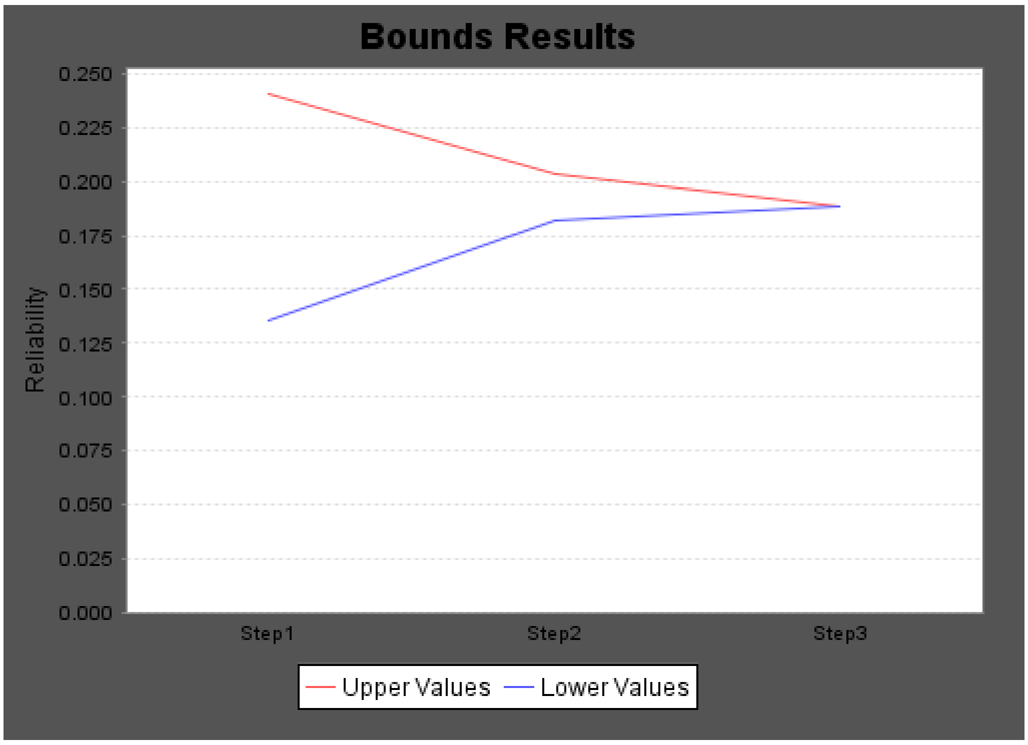
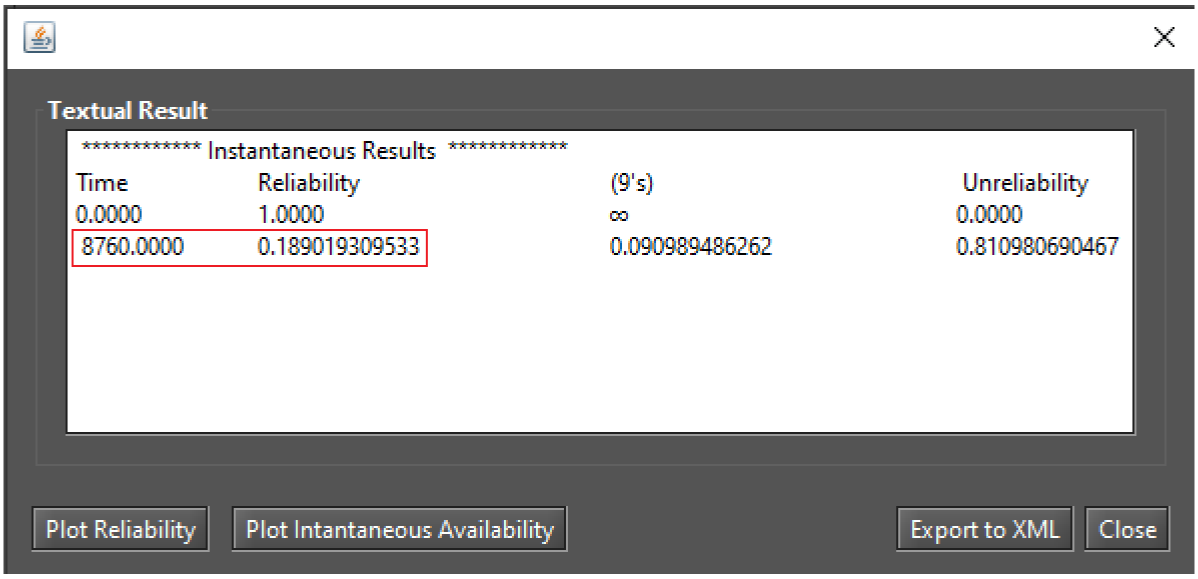


Figure 237: Plotting the Bounds Results

The method adopted to find successive values and the number of iterations (values) is defined by the number of paths and cuts of the model. Increasing the number of iterations, the value found will be closer to the exact value. When the calculation is done for the last path or last cut, the exact value for the metric is found. The exact value will be the value found in the last step. We have performed the exact evaluation for obtaining reliability at 8760h. Figure 238 shows that the reliability at 8760h is equal to the values obtained in the bounds evaluation for the maximum number of steps, as we can see in Figure 236.



```
************ Instantaneous Results ************
Time          Reliability          (9's)               Unreliability
0.0000        1.0000               ∞                   0.0000
8760.0000     0.189019309533       0.090989486262      0.810980690467
```

Figure 238: Reliability Result at 8760h

7.2.4 Component Importance and Total Cost of Acquisition

Component importance is a metric that indicates the impact of a particular component in the system's overall. With these importance values, the most important component (i.e., that one with the highest importance) should be improved to increase system reliability or availability. This evaluation can assist in maintenance actions.

The Component Importante Measures functionality is responsible for identifying the relative importance of each component with respect to the overall system reliability or availability. To use the feature, users should select the "Importance Measures" in the menu Evaluate -> FT Evaluation. After that, they should select one metric of interest on the "Component Importance Measures" dialog box and click on the "Evaluate" button (see Figure 239). If the parameter "Cost" was entered in the events' properties, it is possible to evaluate the relationship between these metrics and investment costs. For reliability metrics, it is necessary to enter the time parameter.

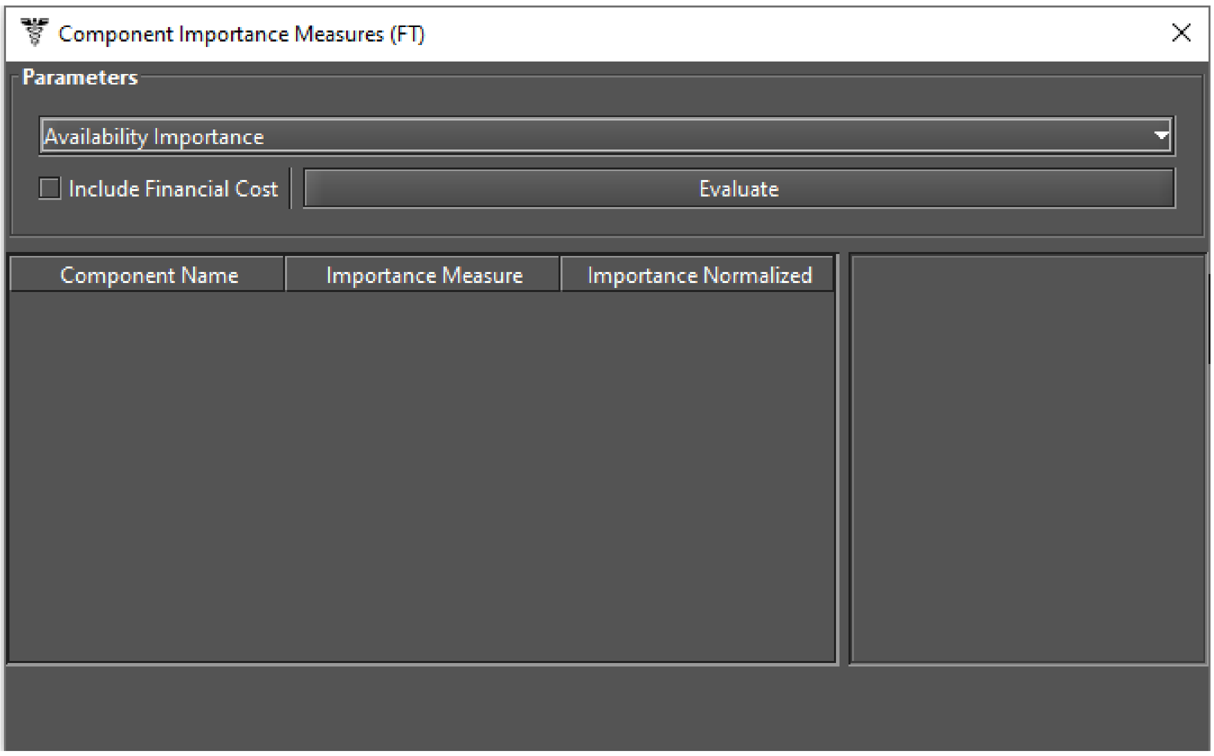


Figure 239: Component Importance Measures

It is possible to compute some importance measures as shown by Figure 240. The measures available are Availability Importance, Reliability Importance (Birnbaum), and Criticallity (Reliability or Availability) Importance (CRI). Last measures can be obtained considering the system's failure "(f)" or functioning.

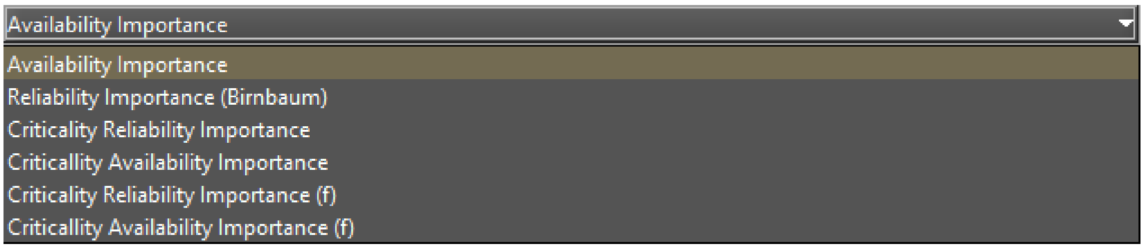

Availability Importance
Availability Importance
Reliability Importance (Birnbaum)
Criticality Reliability Importance
Criticallity Availability Importance
Criticality Reliability Importance (f)
Criticallity Availability Importance (f)

Figure 240: Component Importance Measures - Metrics of Interest

In the following we will demonstrate how to perform the "Component Importance" evaluation. We have performed this evaluation by considering the "Reliability Importance (Birnbaum)" metric for the model presented by Figure 241.

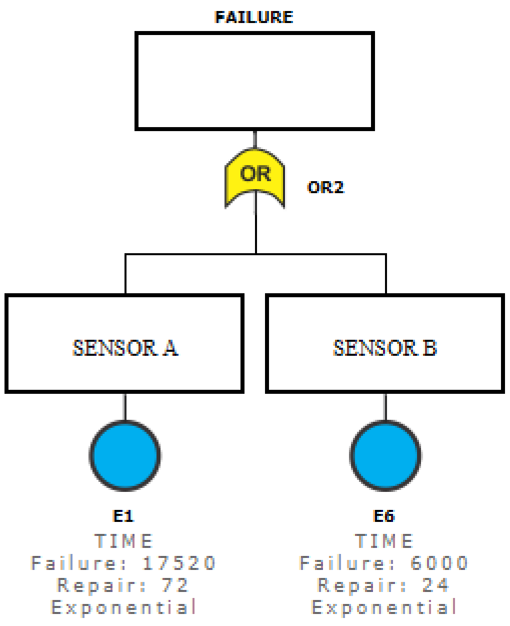
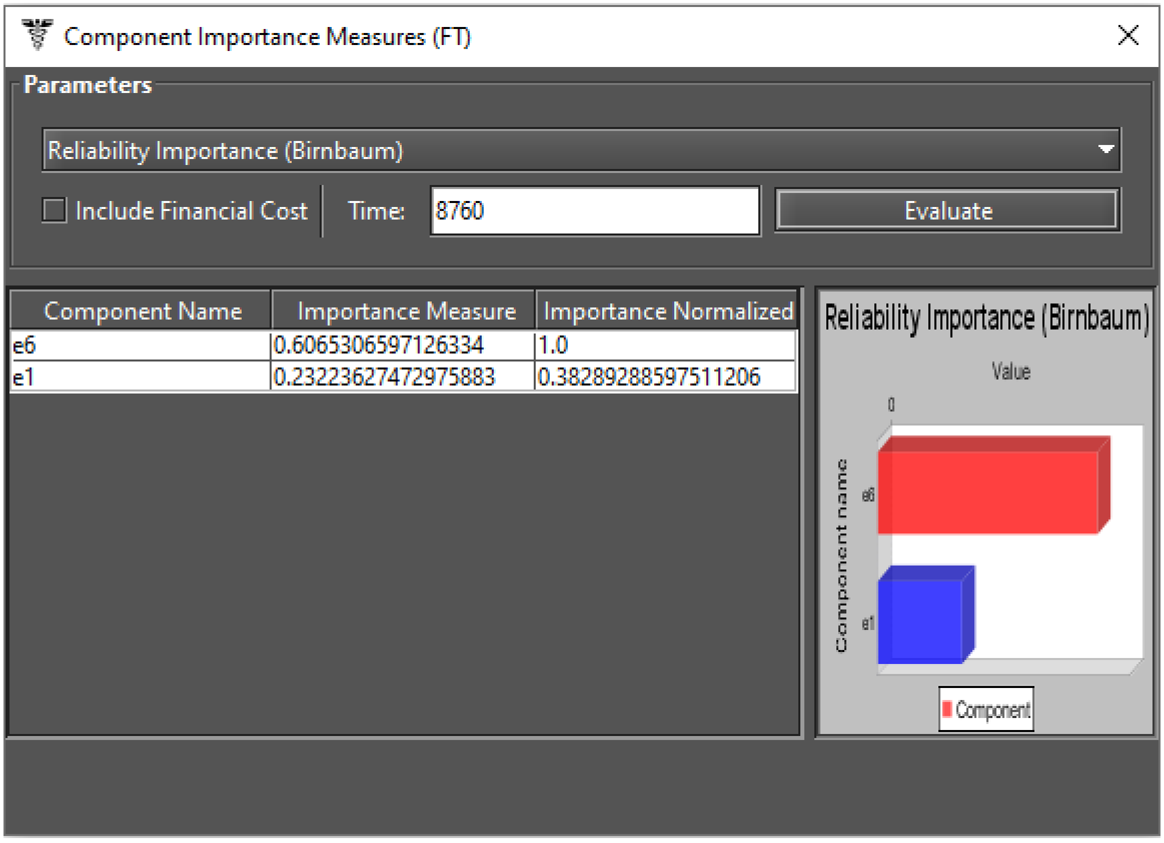


Figure 241: FT Model for Component Importance Evaluation

As we can see by looking at the figure, the system has two sensors and each one is attached with the only gate of the model. In order to the system to failure, it is necessary that at least one of the sensors failures. "Sensor A" has MTTF equal to 17520h and MTTR equal to 72h. "Sensor B" has MTTF equal to 6000h and MTTR equal to 24h. All times are exponentially distributed. When performing the "Component Importance" evaluation by considering the time parameter equal to 8760h, we have obtained the results presented in the Figure 242.


Component Importance Measures (FT)

Parameters

Reliability Importance (Birnbaum)

Include Financial Cost: ☐ Time: 8760

Component Name	Importance Measure	Importance Normalized
e6	0.6065306597126334	1.0
e1	0.23223627472975883	0.38289288597511206

Figure 242: Reliability Importance Results

The results have shown the importance value to each component and a graphical view as a ranking highlighting those most significant in the analysis. As we can see, the component that most impact the system's overall reliability is the "Sensor B" (e6). By replacing this component with another one having the same MTTR, but an MTTF equal to 12000h (see Figure 243), the results have changed as depicted by Figure 244.

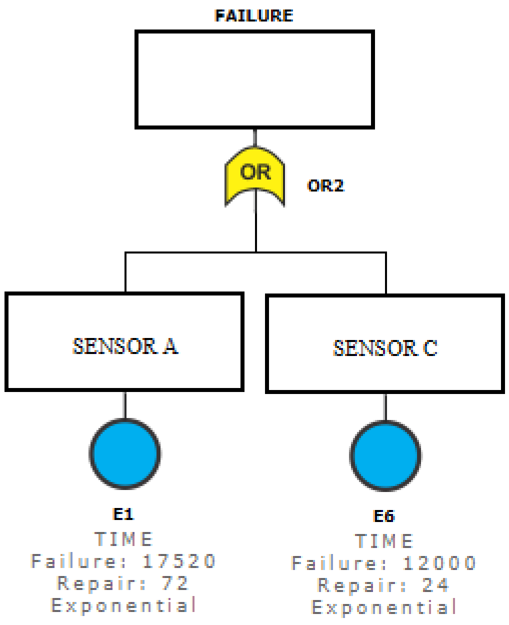
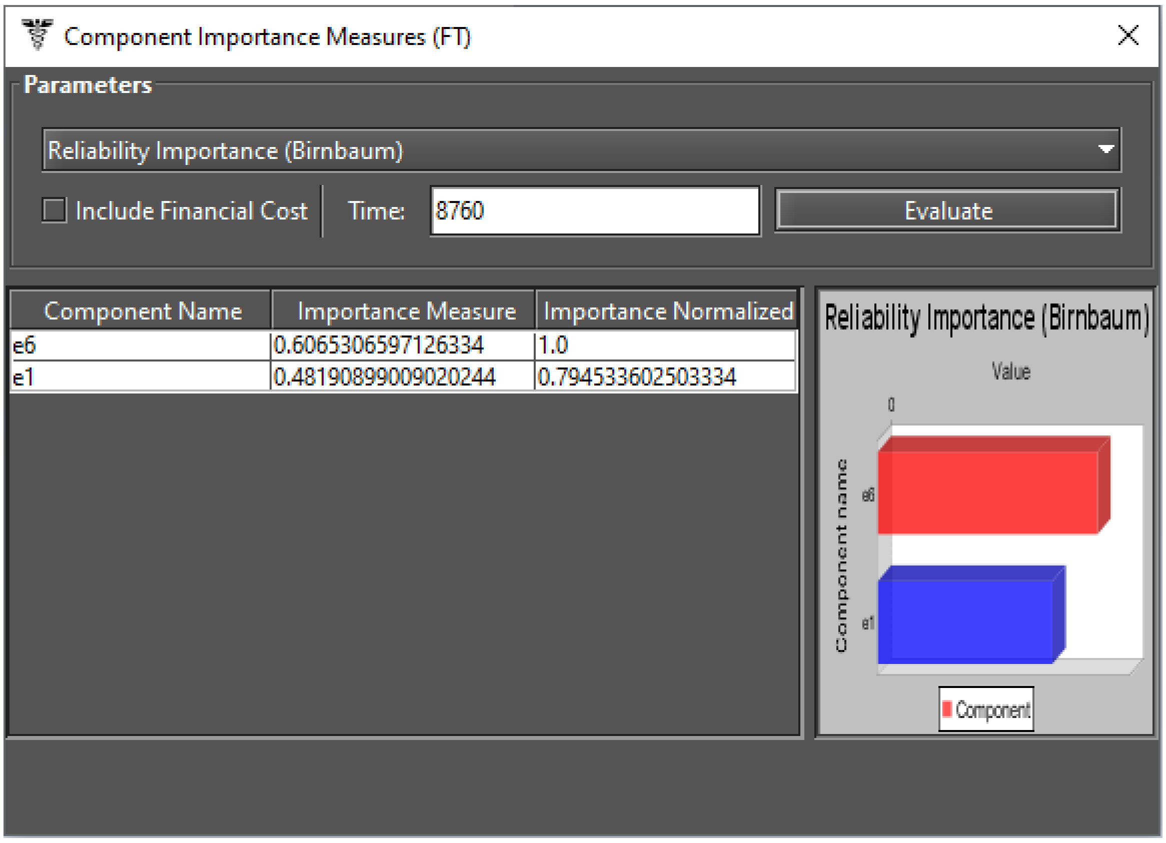


Figure 243: FT Model for Component Importance Evaluation


Component Importance Measures (FT)

Parameters

Reliability Importance (Birnbaum)

☐ Include Financial Cost | Time: 8760 | Evaluate

Component Name	Importance Measure	Importance Normalized
e6	0.6065306597126334	1.0
e1	0.48190899009020244	0.794533602503334

Figure 244: Reliability Importance Results by Considering the Sensor Replacement

Now, we can see that the criticality of the “Sensor A” (e1) has increased. This means that, considering a time interval of 8760h, “Sensor B” is more critical to the overall reliability of the system when compared to “Sensor C”.

7.2.5 Structural and Logical Functions

The Mercury tool can generate the structural and logical functions of the FT models. Both functions are a behavioral representation of the system represented by the model. By applying them, it is possible to evaluate the impact of the failed components on the system status. Structural and logical functions are functions related to the states of the components. The system and each component can be in the working (default) or failed state. The system state is a binary random variable that is determined by the states of the components. If the states of the components are known, the state of the system is also known.

The state can be toggled via the "Properties" dialog box (see Figure 245) and when the state of the event node is failed, the component is depicted by a broken icon, as we have aforementioned.

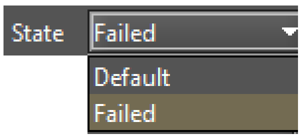


Figure 245: Node State

Now, let us demonstrate how to get these functions by using the Mercury tool. Figure 246 shows an FT model with an AND gate and an OR gate. As we can see, there is a failed node in this FT (event e3), and this node is attached to the OR gate.

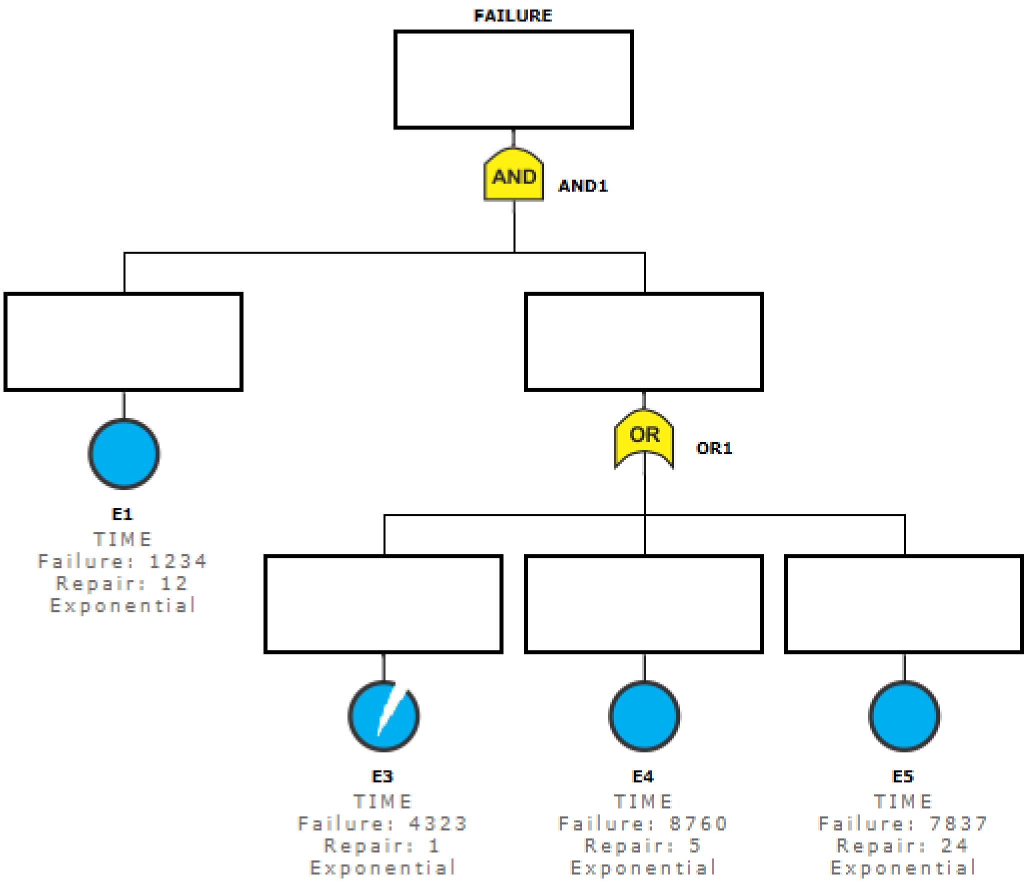
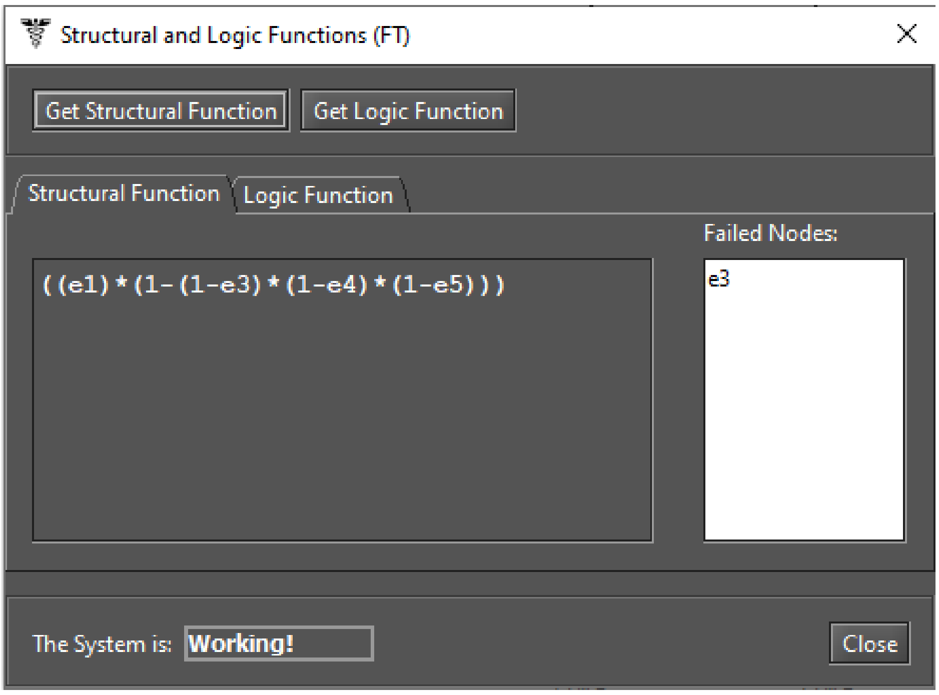


Figure 246: FT Model

The structural and logic functions are accessible in the menu Evaluate -> FT Evaluation -> Get Functions. Figures 247 and 248 show the structural and logic functions of the FT model depicted above, respectively. In addition to the generated expressions, the tool indicates the event nodes marked as failed (inoperative) and the current state of the system. As we can see, the failed node (event e3) does not impact the system's operation.

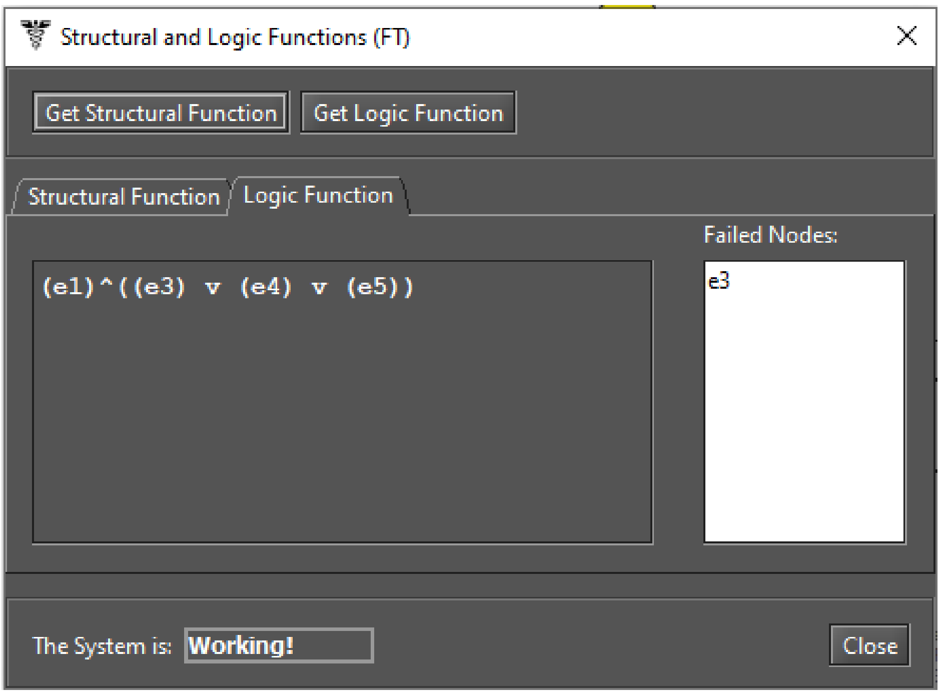


```
((e1)*(1-(1-e3)*(1-e4)*(1-e5)))
```

Failed Nodes: e3

The System is: Working!

Figure 247: Structural Function



```
(e1)^((e3) v (e4) v (e5))
```

Failed Nodes: e3

The System is: Working!

Figure 248: Logic Function

On the other hand, we can see that by changing the state of the event node e1 to failed (see Figure 249), the system status is now failed (see Figure 250).

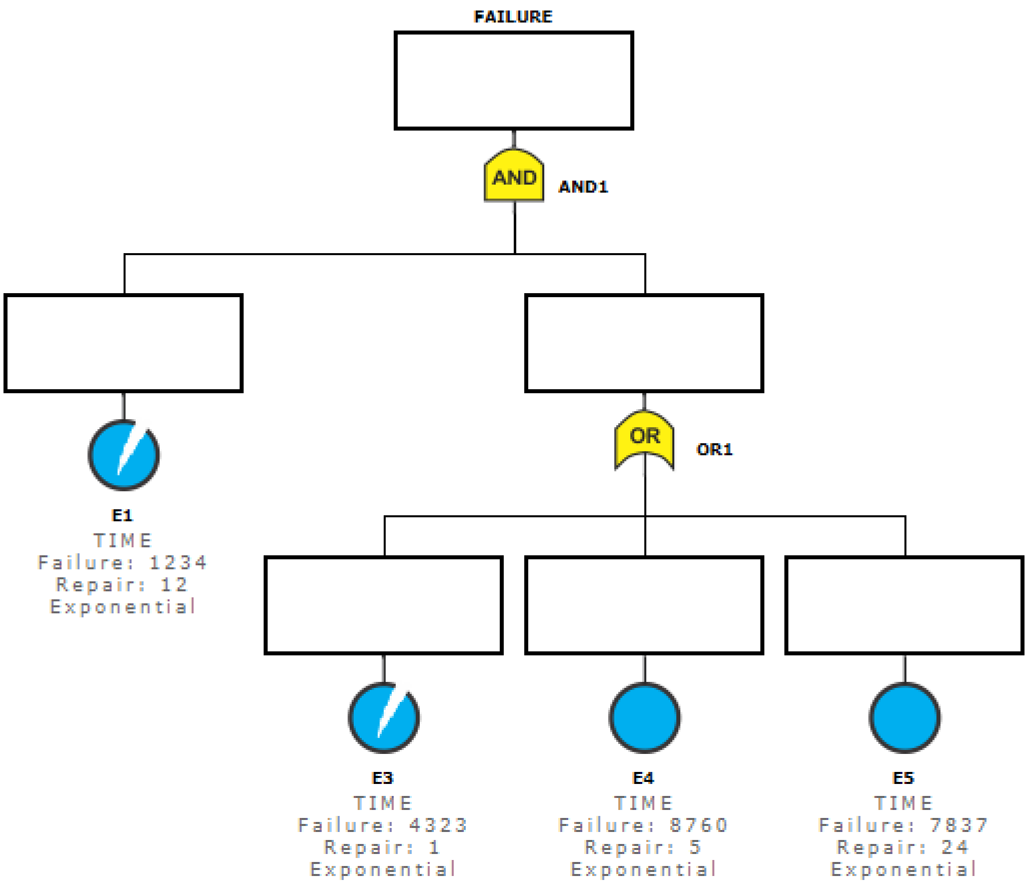
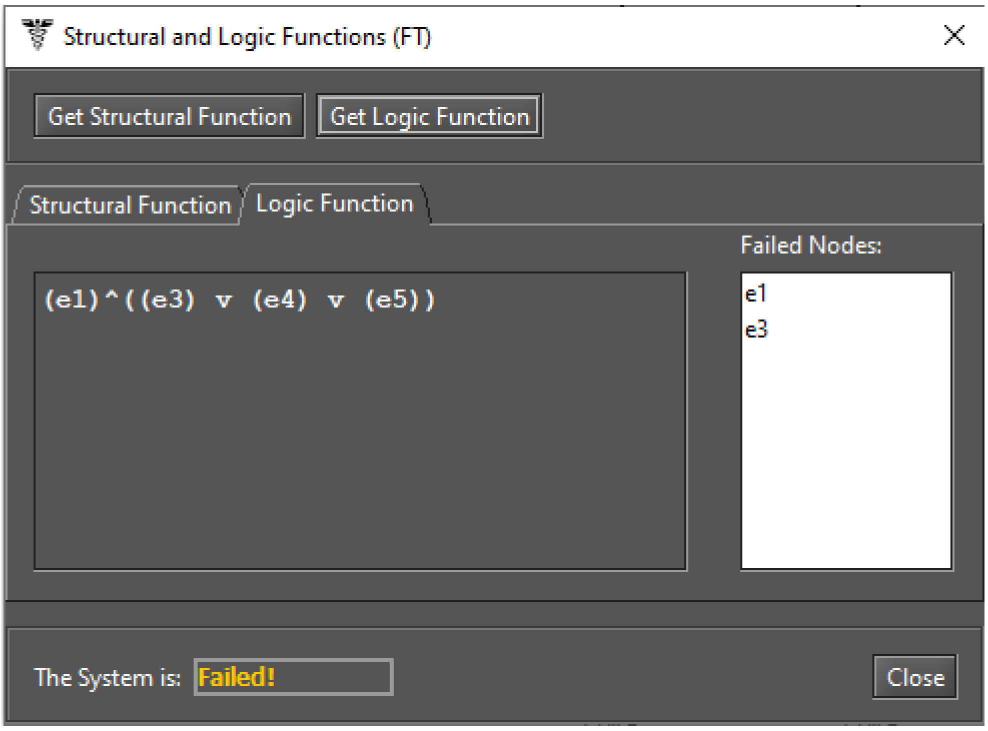


Figure 249: FT Model



```
(e1)^((e3) v (e4) v (e5))
```

Failed Nodes:

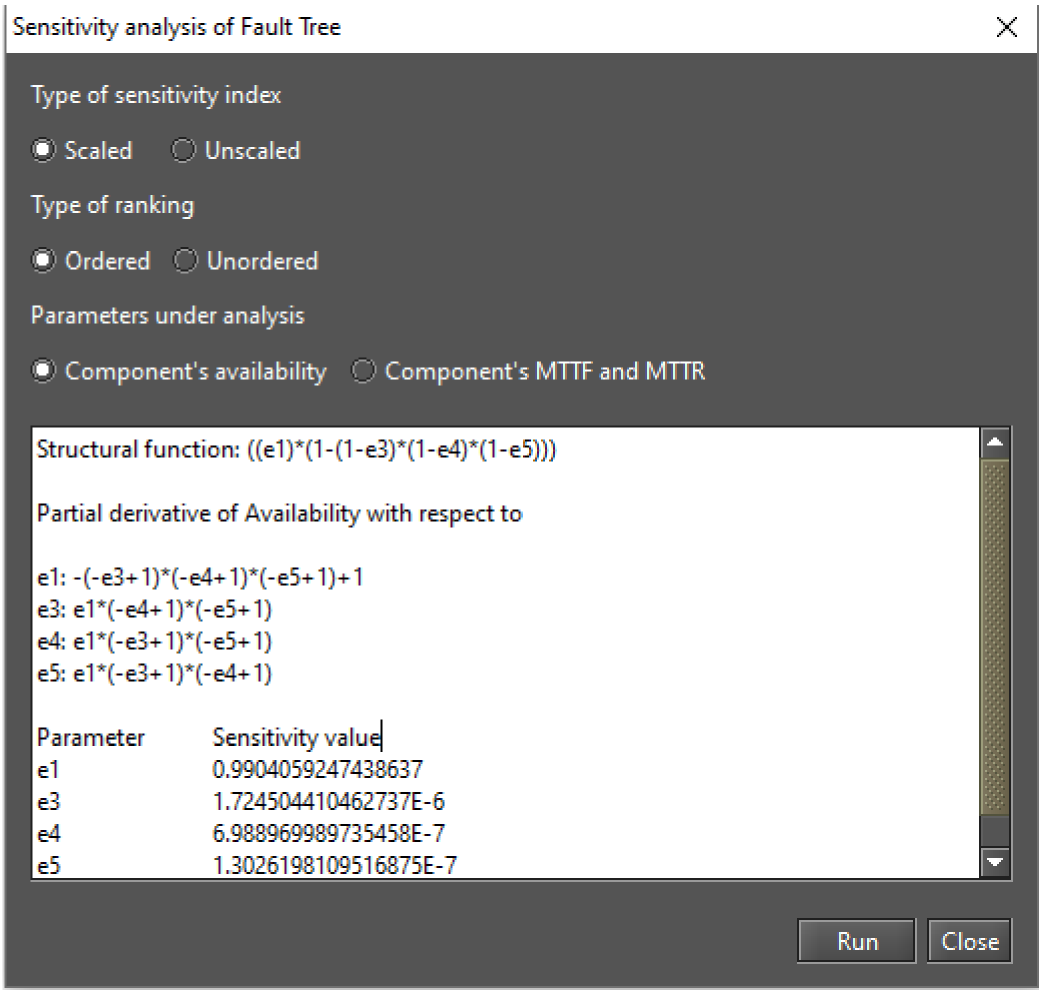
e1
e3

The System is: Failed!

Figure 250: Logic Function and the System State as Failed

7.2.6 Sensitivity Analysis

The Mercury tool provides a functionality named Sensitivity Analysis that enables computing partial derivative sensitivity indices for FTs. Those indices indicate the impact that every input parameter has on the model's availability. If the model has hierarchical nodes, the input parameters of the CTMC sub-model are also considered in the Sensitivity Analysis. This function is acessible in the menu Evaluate -> FT Evaluation -> Sensitivity Analysis. Figure 251 shows the "FT Sensitivity Analysis" window, which presents the partial derivative of the structural equation to every parameter, as well as the sensitivity indices.



Type of sensitivity index

[✓ Scaled] Unscaled

Type of ranking

[✓ Ordered] Unordered

Parameters under analysis

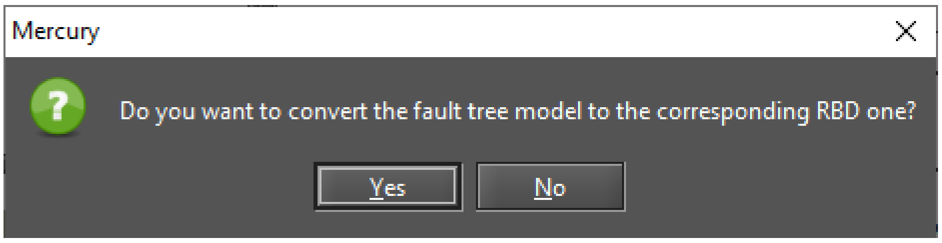
[✓ Component's availability] Component's MTTF and MTTR

```
Structural function: ((e1)*(1-(1-e3)*(1-e4)*(1-e5)))

Partial derivative of Availability with respect to

e1: -(-e3+1)*(-e4+1)*(-e5+1)+1
e3: e1*(-e4+1)*(-e5+1)
e4: e1*(-e3+1)*(-e5+1)
e5: e1*(-e3+1)*(-e4+1)

Parameter        Sensitivity value
e1               0.9904059247438637
e3               1.724504410462737E-6
e4               6.988969989735458E-7
e5               1.3026198109516875E-7
```

Figure 251: Sensitivity Analysis of Fault Tree

7.2.7 Export to RBD model

The Mercury tool provides a feature for converting FTs to RBDs. This feature is accessible in the menu Evaluate -> FT Evaluation -> Export to RBD model. By clicking on it, the user needs to confirm the convertion as depicted by Figure 252. After that, a window is displayed for the user to choose the location and name of the file to be created. In order to see the RBD model, it is necessary to open the created file.


Mercury

Do you want to convert the fault tree model to the corresponding RBD one?

Figure 252: Converting FT to RBD

Figure 253 shows an FT model as an example to be converted, and Figure 254 shows it converted in its respective RBD.

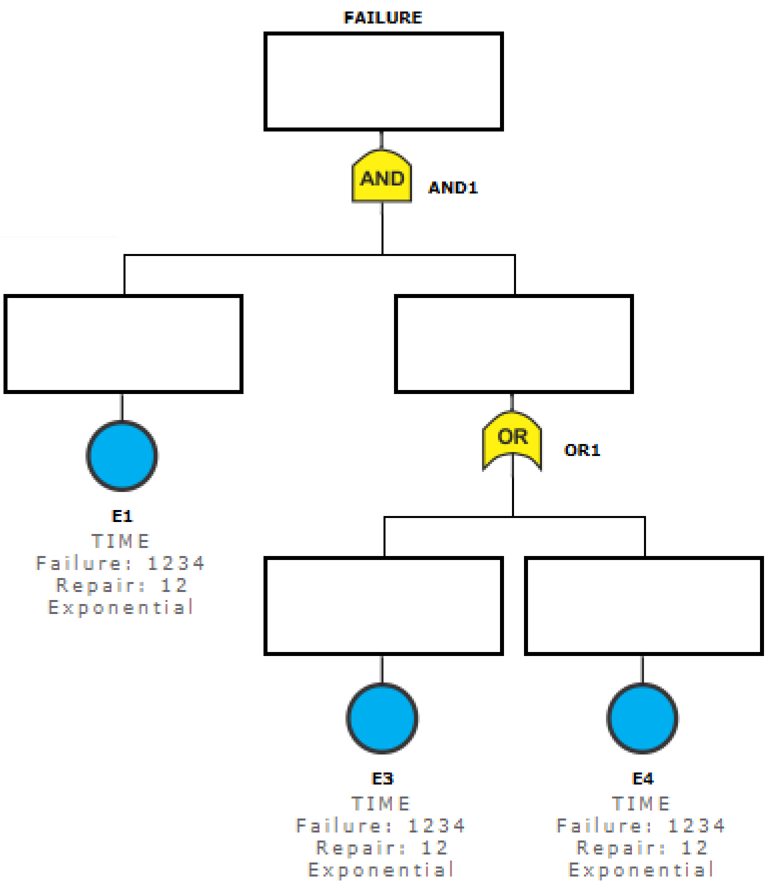


Figure 253: FT Model

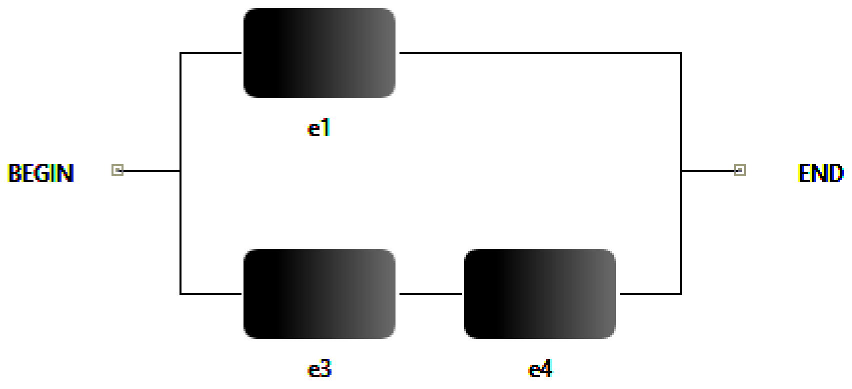


Figure 254: RBD Representing the FT Model

8 Mercury Scripting Language

8.1 Introduction

The Mercury scripting language was designed to allow greater flexibility in model evaluation. To run Mercury scripts, we can use a command-line interface (CLI) tool or the “Script” menu inside the graphical interface. The advantage of using this language in conjunction with the CLI tool is the possibility to automate the project workflow, evaluate models, extract metrics and generate reports and graphs automatically.

In addition, there are other advantages offered by the language, which are not supported by the graphical interface.

- Increased support to hierarchical modeling: any model can call another model and use its results as an internal parameter;
- Increased support for symbolic evaluation and experiments. Parameters of a model can be defined as variables left open. We can change these variables and reevaluate the model to measure the impact of these parameters on certain metrics;
- Support for Petri net transitions with a phase-type delay. This family of distributions can be used to approximate any distribution that does not fit an exponential distribution;
- Support for hierarchical transitions in SPN models. This type of transition can be used to reduce the complexity of models or to express a recurring structure in the model to be reused more easily. Some tools [1] [13] support hierarchical SPN models only for Coloured Petri Nets.

8.2 Script Structure

We define the syntax of a script using BNF notation as follows:

Listing 9: Grammar for a Mercury Script

```
<script> ::= <models> <main_block>

<models> ::= <model> <models> | <model>

<model> ::= <SPN_model> | <CTMC_model> | <RBD_model>
```

A script consists of a section for declaring models, which contains one or more models of the type: CTMC (continuous-time Markov chain), RBD (reliability block diagram), or SPN (stochastic Petri net). Support for FT and EFM formalisms will be included in the next version. At the end of the model section, there is the main block, which has the following syntax:

Listing 10: Grammar for the Main Block

```
<main_block> ::= <main_block> "{"
<main_statements>
"}"

<main_statements> ::= <statement> ";" <main_statements> |
                      <statement> ";"

<statement> ::= <print_statement> |
                <attribution_statement> |
                <for_statement>
```

In the main block, we can modify variables, solve models and print the results obtained. We modify variables to define parameters for a model and collect metric results by using the **solve** function. The “for” command has been included to allow the execution of experiments. With this command, we can modify a variable according to a list of values.

In Figure 255, we show a CTMC model, and in Listing 11, we present the corresponding Mercury script. First, we define a CTMC model named **CTMCModel**, declare its states, transitions, and metrics. Transition rates are defined as function of the parameters **lambda** and **mu**. In the main block, we define values for these parameters and evaluate the metric “m1” of the CTMC model. The result is stored in the variable named **aval**. Finally, we print the content of this variable by using the command **println**.

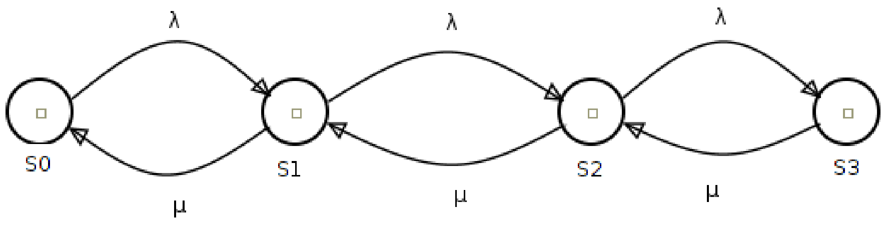


Figure 255: CTMC Model Example

Listing 11: CTMC Model

```
CTMC CTMCModel{
   state S0;
   state S1;
   state S2;
   state S3;

   transition S0 -> S1( rate = lambda);
```

```
    transition S1 -> S0( rate = mu);

    transition S1 -> S2( rate = lambda);

    transition S2 -> S1( rate = mu);

    transition S2 -> S3( rate = lambda);

    transition S3 -> S2( rate = mu);


    metric m1 = stationaryProbability( st = S0 );
}


main{
    lambda = 0.00001;
    mu = 0.01;


    aval = solve( model = CTMCModel, metric = m1 );
    println(aval);
}
```

8.2.1 Reserved Words

In Table 2, we list the reserved words of the language.

state	transition	rate	markov	up
RBD	block	hierarchy	series	parallel
top	model	MTTF	MTTR	main
print	println	for	in	out
metric	solve	value	SPN	SubNet
place	timedTransition	immediateTransition	substitutionTransition	tokens
subnet	inputs	outputs	delay	inhibitors
weight	priority	enablingFunction	serverType	

Table 2: Reserved Words

These words cannot be used as identifiers (of models, variables, functions), metrics, user-defined functions, or as keys in a dictionary structure. For example, for the stationary probability of CTMC metrics, we use the key "st" to inform the state that we want to evaluate the probability. We cannot use the word "state", because this is a reserved word, used for declaring states in a CTMC model.

In the following sections, we describe the syntax for each supported formalism: continuous-time Markov chains, reliability block diagrams, and stochastic Petri nets.

8.3 Continuous Time Markov Chain

In Listing 12, we describe the syntax for declaring CTMC models. A CTMC model contains definitions of states —with the reserved word *state*—, transitions —with the reserved word *transition*—, and metrics. A state can also receive an annotation *up* after the identifier, for availability models. This annotation defines states where the system is considered operational.

Listing 12: Grammar for CTMC Models

```
<CTMC_block> ::= "CTMC" "{"
<ctmc_statemnts>
"}"


<ctmc_statements> ::= <ctmc_statement> ";" <ctmc_statements> |
                              <ctmc_statement> ";"


<ctmc_statement> ::= <state_statement> |
                     <transition_statement> |
                     <metric> ;


<state_statement> ::= "state" <identifier> |
                            "state" <identifier> "up"


<transition_statement> ::= "transition" <identifier> "->" <identifier>
                     "(" "rate" "=" <numeric_exp> ")"



<metric> ::= "metric" <identifier> = <metric_name> "(" <metric_parameters> ")" |
                                      <metric_name>
```

The supported metrics for CTMC models are: i) availability; ii) reward rate for states; iii) stationary probability; and iv) transient probability.

8.3.1 Availability

The availability metric does not take any parameter. This metric returns the sum of all stationary probability for states annotated with the *up* keyword. In the following script, we show an availability model for a redundant private cloud manager.

Listing 13: Availability Metric for a CTMC Model

```
markov RedundantGC{
    state fu up;
    state fw;
    state ff;
    state uf up;
    state uw up;

    transition fw -> fu(rate = sa_s2);
    transition fu -> ff(rate = lambda_s2);
    transition ff -> uf(rate = mu_s1);
    transition uf -> uw(rate = mu_s2);
    transition uw -> fw(rate = lambda_s1);

    transition fw -> uw(rate=mu_s1);
    transition uw -> uf(rate=lambdai_s2);
    transition uf -> ff(rate=lambda_s1);

    transition fw -> ff(rate=lambdai_s2);
    transition fu -> uw(rate=mu_s1);

    metric aval = availability;
}
```

8.3.2 Reward Metric

This metric computes the sum of rates associated with each state. The parameters defined to this metric are a list of pairs: $< state\_name >=< value >$. The metric computes the sum of products of each rate and the stationary probability associated with the state. The states that do not receive any rate, are associated with a rate zero, implicitly.

Below, we list an example of a model with a reward metric. The reader is suggested to check that metrics m1 and m3 produce the same result.

Listing 14: Reward Metric for a CTMC Model

```
markov Teste{


    state s1 up;
    state s2 up;
    state s3;


    transition s1 -> s2 (rate = lambda);
    transition s2 -> s3 (rate = lambda);
    transition s3 -> s2 (rate = mu);
    transition s2 -> s1 (rate = mu);


    metric m1 = availability;
    metric m2 = reward ( s1 = 1/5, s2 = 1/4, s3 = 1/3 );
    metric m3 = reward ( s1 = 1, s2 = 1 );
}
```

8.3.3 Stationary and Transient Probabilities

The most recurrent metrics used with CTMCs are stationary and transient probabilities associated with states. The stationary probability of a state *S* corresponds to the proportion of time that the model remains in this state. The transient probability of a state *S* within a time *T*, corresponds to the probability to be in this state *S*, after *T* units of time from initial time (t = 0).

In the Mercury language, we use the **stationaryProbability(st = S)** metric to obtain the stationary probability associated with a state *S*. For transient probability, we need also to provide the time *T*, and initial probability for each state. That corresponds to the probability that at time T = 0, the model will be in that state. In the scripting syntax, the states that are not declared in the list of initial probabilities, receive an initial probability equal to 0. It is important to highlight that the sum of all initial probabilities must be equal to 1, otherwise, an exception will be raised.

The following we illustrate how to get the metrics for stationary and transient probabilities taking into account a CTMC model as an example.

Listing 15: Stationary and Transient Metrics for a CTMC Model

```
markov Test3{
    state s0;
    state s1;
    state s2;
    state s3;
    state s4;

    transition s0 -> s2 (rate = a);
    transition s2 -> s1 (rate = b);
    transition s1 -> s4 (rate = a);
    transition s2 -> s3 (rate = b);

    transition s3 -> s4 (rate = c);
    transition s4 -> s0 (rate = c);

    metric m1 = reward( s0 = 1, s1 = 2 );
    metric m2 = stationaryProbability ( st = s2 );

    metric t0 = transientProbability (
        time = 100,
        st = s0,
        initialProbabilities = ( s0 = 0.5, s3 = 0.5 )
    );
}
```

8.4 Reliability Block Diagram

An RBD model is composed of:

- Exponential blocks that represent components with an associated mean time to failure and mean time to repair parameters;
- Hierarchical blocks that are evaluated by calling other external models;
- Series/parallel arrangements of other blocks;
- Declaration of the top-level block;
- RBD metrics.

Listing 16 shows the grammar for RBD models.

Listing 16: RBD Grammar

```
<RBD_model> ::= "RBD" "{" <rbd−statements> "}"

<rbd−statements> ::= <rbd_statement> ";" <rbd_statements>|
                         <rbd_statement> ";"

<rbd_statement> ::= <block_statement>
         | <series_block_statement>
         | <parallel_block_statement>
         | <top_block_statement>
         | <rbd_metrics>

<block_statement> ::= <exp_block_statement> |
                     <hierarchy_block_statement>

<exp_block_statement> ::= "block" <identifier>
     "(" "MTTF" "=" <numeric_exp> ","
         "MTTR" "=" <numeric_exp> ")"

<hierarchy_block_statement> ::= hierarchy <identifier> "("
                 "availability" "=" <numeric_expression> ") ";" |
                  hierarchy <identifier> "("
                 "reliability" "=" <numeric_expression> ") ";"



<series_block> ::= "series" <identifier> "(" <identifire_list> ")"  ";"

<parallel_block> ::= "parallel" <identifier> "(" <identifire_list> ")"  ";"

<top_block> ::= "top" <identifier> ";"
```

We have four metrics available to RBD models:

- Availability;
- Mean time to failure (MTTF);
- Mean time to repair (MTTR);

- Reliability.

The three first metrics do not take any parameters: steady-state availability, MTTF, and MTTR. On the other hand, reliability and instantaneous availability metrics require a *time* parameter.

By considering the model depicted in Figure 256, we generated its script definition as shown in Listing 17.

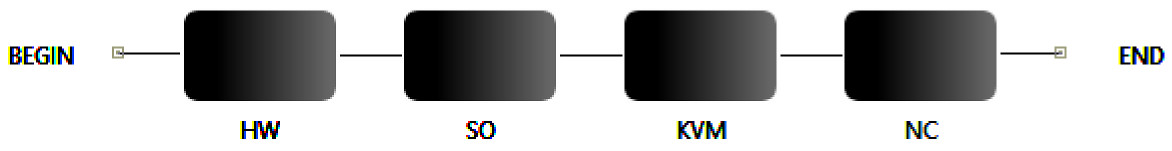


Figure 256: RBD Representing a Cloud Node [4]

Listing 17: RBD Script

```
t = 100;

RBD Model{
    block HW( MTTF = 4000.0, MTTR = 72.0);
    block SO( MTTF = 2500.0, MTTR = 12.0);
    block KVM( MTTF = 4000.0, MTTR = 24.0);
    block NC( MTTF = 4000.0, MTTR = 24.0);
    series s0(HW, SO, KVM, NC );

    top s0;

    metric av = availability;
    metric rel = reliability( time = t );
    metric mttf = mttf;
    metric mttr = mttr;
}

main{
          av = solve(Model, av);
          rel = solve(Model, rel);
          mttf = solve(Model, mttf);
          mttr = solve(Model, mttr);

          println("Availability: " .. av );
          println("Reliability: " .. rel );
```

```
        println("Mean time to failure: " .. mttf );
        println("Mean time to repair: " .. mttr );
}
```

8.5 Stochastic Petri Nets

In the Mercury scripting language, a Petri Net is described in terms of places and transitions. Places can be defined with an (optional) initial marking. Transitions can be of three types: immediate, timed, and substitution. "Substitution" transitions allow us to create modular and reusable Petri nets. This functionality is available only in the scripting language. Another exclusive feature of the scripting language is the support for *phase-type* distributions. In this section, we will show a simple SPN as an example containing only exponential and immediate transitions.

Listing 18 shows the grammar in BNF notation for describing SPNs models in the Mercury language. Basically, we have three distinct statements: place statements, transition statements, and metric statements. The arcs connecting transitions and places are defined inside the transitions, as parameters: *inputs*, *outputs*, and *inhibitors*. Timed transitions have as parameters the delay and the server type. If ommited, the "server type" parameter will be "SingleServer" by default. Immediate transitions have as parameters (optional): weight, priority, and an enabling function. The metrics are defined in terms of a string representing a reward metric, the same used in the graphical interface.

Listing 18: Grammar for SPN Models

```
<SPN–Model>   ::= "SPN" "{"
<spn_statements>
"}"

<spn_statements> ::= <spn_statement> ";" <spn_statements> |
                         <spn_statement> ";"

<spn_statemnt> ::= <place_statement> |
                         <transition_statement> |
                         <metric_statement>

<place_statement> ::= "place" <identifier> |
                         "place" <identifier> "(" <numeric_exp> ")"

<transition_statement> ::= <timed_transition> |
                                   <immediate_transition> |
                               <substitution_transition>
```

```
<timed_transition> ::=  "timedTransition" <identifier> "{"
                              [ "inputs" "=" "(" <arc_list> ")" "," ]
                              [ "outputs" "=" "(" <arc_list> ")" "," ]
                              [ "inhibitor" "=" "(" <arc_list> ")" "," ]
                              [ "serverType" "=" { "SingleServer" |
                              "InfiniteServer" } "," ]
                              [ "delay" "=" <delay_exp> "," ]
"}"

<immediate_transiton> ::=  "immediateTransition" <identifier> "{"
                              [ "inputs" "=" "(" <arc_list> ")" "," ]
                              [ "outputs" "=" "(" <arc_list> ")" "," ]
                              [ "inhibitor" "=" "(" <arc_list> ")" "," ]
                              [ "weight" "=" <numeric_exp> "," ]
                              [ "priority" "=" <numei_exp> "," ]
"}"
```

To illustrate the syntax for modeling SPNs using the Mercury scripting language, we have proposed a model for an M/M/1/K queue, based on [14]. That model is depicted by Figure 257.

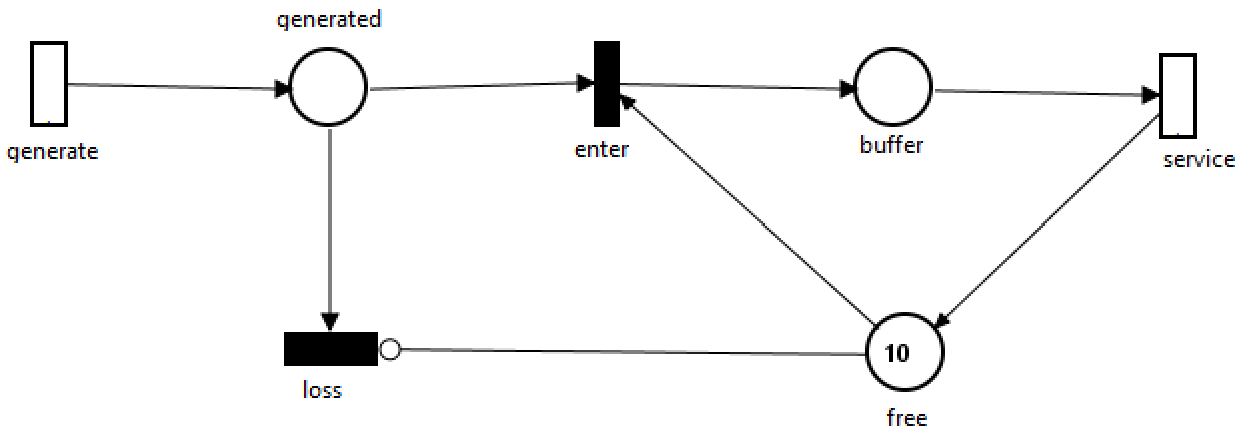


Figure 257: SPN Model Representing an M/M/1/k Queue[14]

Transition *generate* creates tokens that correspond to service requests. Each generated token is placed in the place *generated* and, thereafter, a choice is made. The token can be queued for processing by the server, if there is a free slot in the queue (tokens in the " free" place). Otherwise, the token is discarded, being represented by the firing of the immediate transition "loss" . The place "free" controls the firing of this transition through an inhibitor arc. The tokens waiting in the queue are placed in the "buffer". The transition "service" represents the processing of the requests. Considering that it is an M/M/1/K queue, we only have one server to handle all requests. Therefore, the server semantic assigned to the “service” transition is SINGLE SERVER.

Listing 19 shows the script for running the stationary analysis on the SPN model previously described. The parameter **method** of the **stationaryAnalysis** function can have only the values “direct” or “iterative”. “Direct” corresponds to the **Direct - GTH (Grassmann-Taksar-Heyman)** method. “Iterative” corresponds to the **Gauss-Seidel** method.

Listing 19: Script for Stationary Analysis

```
k = 10;
mu = 2;
lambda = 1;

SPN Model{
        place buffer;
        place free( tokens= 10 );
        place generated;

        immediateTransition enter(
                inputs = [generated, free],
                outputs = [buffer]
        );

        immediateTransition loss(
                inputs = [generated],
                inhibitors = [free]
        );

        timedTransition generate(
                outputs = [generated],
                delay = lambda
        );

        timedTransition service(
                inputs = [buffer],
                outputs = [free],
                delay = mu
        );
        metric m1 = stationaryAnalysis( method = "direct",
            expression = "P{#buffer>0}" );
}

main {
        setIntegerParameters("k", "mu", "lambda");

        m1 = solve( Model,m1 );
        println(m1);
}
```

Listing 20 shows the script for running the stationary simulation on that model. Below, we describe each parameter of the **stationarySimulation** function.

- **confidenceLevel.** The confidence interval for obtaining the metrics.
- **maxRelativeError.** Defines the maximum relative error that is one of the stop conditions of the simulation.
- **minFiringTransitions.** Sets the minimum number of firings for each transition in the model. This number of firings is another condition for stopping the simulation. When set to 0, the simulator does not consider the number of firings in order to stop the simulation. Entering a value greater than 0, when the number of firings for each transition is equal to the defined value the simulation stops.
- **warmup.** Sets the minimum warm-up period. The warm-up phase is the period when the model is not considered to be in a steady-state, and the metrics are not collected in that period. There are some methods for estimating whether the model has entered a stationary phase, but Mercury requires the user to define a value for the warm-up time. As we are evaluating stochastic models, it is expected that the warm-up time is not an exact value for each simulation performed. Therefore, the user defines a minimum warm-up time and, once the global simulation time is equal to or greater than the warm-up time defined by the user, the simulation begins to collect the metrics, generate the batches, and calculate statistics.
- **batchsize.** Defines the number of samples that will constitute each batch in the simulation.
- **maxTimeMilliseconds.** It is used to define the maximum simulation time. This time corresponds to the physical time and must be defined in seconds. When set to 0, the simulator does not consider this parameter. If one of the stop conditions is not met before this maximum time is reached (maximum relative error or number of firings for each transition), then the simulation stops when this time is reached.

Listing 20: Script for Stationary Simulation

```
k = 10;
mu = 2;
lambda = 1;

SPN Model{
        place buffer;
        place free( tokens= 10 );
        place generated;

        immediateTransition enter(
                inputs = [generated, free],
                outputs = [buffer]
        );

        immediateTransition loss(
                inputs = [generated],
                inhibitors = [free]
        );

        timedTransition generate(
                outputs = [generated],
                delay = lambda
        );

        timedTransition service(
                inputs = [buffer],
                outputs = [free],
                delay = mu
        );

        metric m1 = stationarySimulation( parameters = ( warmup=0,
                        confidenceLevel=90,
                        maxRelativeError=0.05,
                        minFiringTransitions = 0,
                        maxTimeMilliseconds=0,
                                    batchSize=30
                                ), expression = "P{#buffer>0}" );
}


main {
          setIntegerParameters("k", "mu", "lambda");


          m1 = solve( Model,m1 );
          println(m1);
}
```

References

[1] R. German, C. Kelling, A. Zimmermann, and G. Hommel, “Timenet: a toolkit for evaluating non-markovian stochastic petri nets,” *Performance Evaluation*, vol. 24, no. 1, pp. 69–87, 1995.

[2] A. Desrochers and R. Al-Jaar, *Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis.* IEEE Press, 1995.

[3] H. Pham, “System reliability concepts,” in *System Software Reliability.* Springer, 2006, pp. 9–75.

[4] J. Dantas, R. Matos, J. Araujo, and P. Maciel, “An availability model for eucalyptus platform: An analysis of warm-standy replication mechanism,” in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on.* IEEE, 2012, pp. 1664–1669.

[5] R. Matos Junior, A. Guimaraes, K. Camboim, P. Maciel, and K. Trivedi, “Sensitivity analysis of availability of redundancy in computer networks,” in *CTRQ 2011, The Fourth International Conference on Communication Theory, Reliability, and Quality of Service.* IARIA, Apr 2011, pp. 115–121. [Online]. Available: http://www.thinkmind.org/index.php?view=article&articleid=ctrq_2011_6_10_10047

[6] R. S. Matos, P. R. M. Maciel, F. Machida, D. S. Kim, and K. S. Trivedi, “Sensitivity analysis of server virtualized system availability,” *IEEE Transactions on Reliability*, vol. 61, no. 4, pp. 994–1006, 2012.

[7] G. Callou, P. Maciel, D. Tutsch, and J. Araujo, “Models for dependability and sustainability analysis of data center cooling architectures,” in *Dependable Systems and Networks (DSN), 2012 IEEE International Conference on*, Jun 2012, pp. 1 –6.

[8] L. Ford and D. R. Fulkerson, *Flows in networks.* Princeton Princeton University Press, 1962, vol. 1962.

[9] J. Ferreira, G. Callou, and P. Maciel, “A power load distribution algorithm to optimize data center electrical flow,” *Energies*, vol. 6, no. 7, pp. 3422–3443, 2013.

[10] J. Ferreira, G. Callou, J. Dantas, R. Souza, and P. Maciel, "An algorithm to optimize electrical flows," in *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics.* IEEE Computer Society, 2013, pp. 109–114.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein *et al.*, *Introduction to algorithms.* MIT press Cambridge, 2001, vol. 2.

[12] R. Bellman, "On a routing problem," DTIC Document, Tech. Rep., 1956.

[13] A. V. Ratzer, L. Wells, H. M. Lassen, M. Laursen, J. F. Qvortrup, M. S. Stissing, M. Westergaard, S. Christensen, and K. Jensen, "Cpn tools for editing, simulating, and analysing coloured petri nets," in *Applications and Theory of Petri Nets 2003.* Springer, 2003, pp. 450–462.

[14] R. German, "A concept for the modular description of stochastic petri nets (extended abstract," in *Proc. 3rd Int. Workshop on Performability Modeling of Computer and Communication Systems*, 1996, pp. 20–24.

A Syntax of CTMC Measures, Parameters, State Names, and State Rewards

Output measures for CTMC models created in the GUI must be defined according to the following notation:

“P{”<state_name>“}” = Probability of being in the state named <state_name>

“R{”<state_name>“}” = Reward rate of the state named <state_name>

“R{}” = Steady-state reward of the system

The formal syntax for output measures, names of states and parameters, as well as for transition rates is defined as follows:

Listing 21: Syntax of Components for CTMC model

```
<output_measure> ::= <output_value>
                  | ‘‘−’’ <output_measure>
                  | ‘‘(’’ <output_measure>‘‘)’’
                  | <output_measure> <num_op> <output_measure>

<output_value> ::= <probability_measure>
                  | <reward_measure>
                  | <real_constant>
                  | <integer_value>

<probability_measure> ::= ‘‘P{’’<state_name>‘‘}’’

<reward_measure> ::= ‘‘R{’’ {<state_name>} ‘‘}’’

<state_name> ::= {<identifier>}+

<parameter_name> ::= {<identifier>}+

<transition_rate> ::= <expression>

<reward_rate> ::= <expression>

<expression> ::= <real_value>
              | ‘‘−’’ <expression>
              |‘‘(’’ <expression>‘‘)’’
              | <expression> <num_op> <expression>
```

```
<num_op> ::= ‘‘+’’ | ‘‘   ’’ | ‘‘∗’’ | ‘‘/’’


<real_value> ::= <parameter_name>
                |<real_constant>
                |<integer_constant>


<real_constant> ::= {<digit>}+‘‘.’’{<digit>}+


<integer_constant> ::= {<digit>}+


<identifier> ::= {letter | digit}+


<letter> ::= ‘‘A’’\textendash‘‘Z’’ | ‘‘a’’\textendash‘‘z’’


<digit> ::= ‘‘0’’\textendash‘‘9’’
```

The basic symbols have the following meanings:

“symbol”: terminal symbol

< symbol > : non-terminal symbol

symbol1 | symbol2 : symbol 1 or symbol2

{symbol}+ one or more occurrences of symbol

symbol1–symbol2 : range of values between symbol1 and symbol2

B Syntax of SPN Metrics, Guard Expressions, and Arc Multiplicity Dependent on Marking.

This section presents the specification related to SPN metrics, guard expressions, and arc multiplicity dependent on marking. We present a formal syntax description by using Backus-Naur Form (BNF).

Three different expressions can be used in the Mercury tool (see Listing 22). SPN expressions are represented as "Metrics", "GuardExpressions", and "MarkingDependentMultiplicites". "Metrics" can be a probability or an expectation and are used to represent the evaluated metrics. "GuardExpressions" are adopted to represent logic expressions in order to enable/disable the firing of transitions. "MarkingDependentMultiplicites" are numeric expressions that are evaluated depending on the current marking to a specific arc multiplicity.

Listing 22: Syntax of Components for SPN model

```
<Metric> ::=       ''P{''<logic_condition>''}''
                 | ''E{''<marking_function>''}''

<MarkingDependentMultiplicity> ::= <if_else_exp>

<GuardExpression> ::= <logic_expression>

<if_exp> ::= {''IF(''<logic_condition>''):(''<expr>'')''}

<if_list> ::= <if_exp> | <if_list> <if_exp>

<if_else_exp> ::= <if_list> +''ELSE(''<expr>'')''
                   | <expr>

<expr> ::= <real_value>
           |''-''<expr>
           |''(''<expr>'')''
           |''(''<expr>'')''<num_op>''(''<expr>'')''

<real_value> ::= <identifier>
                 | <real_const>
                 | <int_value>

<real_const> ::= {<digit>}+''.''{<digit>}+
<logic_condition> ::= <comp>
                    | ‘‘(’’<logic_condition>‘‘)’’
                    | ‘‘NOT(’’<logic_condition>‘‘)’’
                    | ‘‘(’’<logic_condition>‘‘)’’AND‘‘(’’<logic_condition>‘‘)’’
                    | ‘‘(’’<logic_condition>‘‘)’’OR‘‘(’’<logic_condition>‘‘)’’

<comp> ::= <mark_function><comp_op><mark_function>

<comp_op> ::= ‘‘/=’’ | ‘‘=’’ | ‘‘<’’ | ‘‘>’’ | ‘‘<=’’ | ‘‘>=’’

<mark_function> ::= ‘‘(’’<mark_function>‘‘)’’<num_op>‘‘(’’<mark_function>‘‘)’’
                    | ‘‘(’’<mark_function>‘‘)’’
                    | <int_value>

<num_op> ::= ‘‘+’’ | ‘‘    ’’ | ‘‘*’’ | ‘‘/’’

<int_value> ::= <int_const>
                    |<identifier>
                    |<mark>

<int_const> ::= {<digit>}+

<identifier> ::= {<letter>|<digit>}+

<letter> ::= ‘‘A’’\textendash‘‘Z’’ | ‘‘a’’\textendash‘‘z’’

<digit> ::= ‘‘0’’\textendash‘‘9’’

<mark> ::= ‘‘#’’<identifier>
```

B.1 GENERAL COMMENTS ABOUT SPN SYNTAX

In this syntax, all the elements of a given expression are separated by parentheses. For instance, suppose we want to evaluate the probability of having more than two tokens on place P1 and zero tokens on place P2. The corresponding expression is:

$$P\{(\#P1 > 2)AND(\#P2 = 0)\}//\text{CORRECT SYNTAX}$$

IMPORTANT. Empty spaces inside the expressions are not allowed. Therefore, the following expression is not allowed.

$$P\{\#P1>2AND\#P2=0\}//\text{WRONG SYNTAX}$$

Generally, guard expressions are composed of different comparisons composed by ANDs, ORs, and NOTs. For instance, let us see the following expression:

$$(\#P1=1)AND(\#P2=2)//\text{CORRECT SYNTAX}$$

This expression can be adopted as an enabling function to allow a transition to fire only if the place P1 has one token and P2 has two tokens. Again, empty spaces inside the expressions are not allowed and all the sub-expressions must be combined by using parentheses.

$$\#P1=1AND\#P2=2//\text{WRONG SYNTAX}$$

Regarding "if-else" expressions. The language supports if-else expressions to represent MarkingDependent-Multiplicity. This component is used to represent the number of tokens in places. When used in language, these expressions may change the place marking based on other place markings. For instance, suppose a model with two places P1 and P2, and the marking of P1 is one if P2 has no tokens and zero if P2 has tokens. In this case, P1 marking should be

$$IF(\#P2=0):(1)ELSE(0)$$

It is also possible to have nested if-else expressions. To explain that, let's extend the previous example and consider that the model has 4 places (P1, ..., P4) and the marking of P1 will be one if P2 has no tokens, zero if P3 has one token, two if P4 has no tokens and three otherwise. The corresponding expression should be.

$$IF(\#P2=0):(1)IF(\#P3=1):(0)IF(\#P4=0):(2)ELSE(3)$$

This expression is similar to nested if, elseif, ..., else expressions in standard programming languages like C or Java.