Virtual Satellite 4 FDIR

User Manual

Version 4.10.0, 2020-04-23T10:01:46Z

Table of Contents

1. What is Virtual Satellite 4 FDIR? . . . 1
2. Purpose . . . 2
3. Getting Started . . . 3
   3.1. Modeling Workflow . . . 3
4. Fault Modeling . . . 4
   4.1. Modeling Fault Trees . . . 4
      4.1.1. Fault Tree Elements . . . 4
      4.1.2. The graphical Fault Tree Editor . . . 8
   4.2. Modeling Detection . . . 8
   4.3. Data exchange with the GALILEO Format . . . 8
5. Recovery Modeling . . . 9
   5.1. Modeling Recovery Automata . . . 9
      5.1.1. Recovery Automata . . . 9
      5.1.2. The graphical Recovery Automaton Editor . . . 9
6. FDIR Analysis . . . 10
   6.1. Configuring Analysis information . . . 10
   6.2. Qualitative Analysis . . . 10
   6.3. Quantitative Analysis . . . 10
   6.4. Using STORM . . . 10
7. FDIR Reporting . . . 11
   7.1. Using the SAVOIR/FDIR Report Template . . . 11
   7.2. Excel Exports . . . 11
8. FDIR Synthesis . . . 12
   8.1. Fault Tree Generation . . . 12
   8.2. Recovery Automata Synthesis . . . 12
Legal - License & Copyright . . . 13

Chapter 1. What is Virtual Satellite 4 FDIR?

This user manual describes how to use the Software Virtual Satellite 4 FDIR (VirSat4 FDIR). The software is an extension to Virtual Satellite 4 CORE, for a guide on how to use Virtual Satellite in general, please refer to the Virtual Satellite 4 CORE Manual.

Chapter 2. Purpose

VirSat4 FDIR focuses on modeling Fault Detection, Isolation, and Recovery. This includes among others: Models on failure behavior, recovery behavior, detection behavior, analysis results, and more. The main purpose of VirSat4 FDIR is to enable the analysis of FDIR concepts by means of mathematically well founded evaluation.

The software primarily follows the ESA ECSS standards by means of the SAVOIR FDIR Handbook (http://savoir.estec.esa.int/SAVOIRDocuments.htm). For clarification on vocabulary, further details on FDIR analysis and process, please check the SAVOIR FDIR Handbook. Accessing the handbook requires registration / login. Please note that this in return is restricted to ESA member states.

Chapter 3. Getting Started

3.1. Modeling Workflow

Learn in this section about the recommended workflow for approaching FDIR modeling & analysis. The workflow for creating fault models is up to the preference of the users. Nevertheless, basing the workflow on the following steps is recommended:

- Create a system model using the product structures concepts (PS Concept). For details check the Virtual Satellite 4 CORE User Manual.
- Create a SubSystem dedicated to Risk / FDIR.
- Create a list of Feared Events in this sub system and assign severity categories to them.
- Create a list of faults, with their basic events, for each equipment.
- Perform a Fault Tree Analysis to determine the fault propagations.
- Create the FDIRParameters category at the top of your system model and configure it.
- Create FDIR analysis categories for the desired faults
- Refine the system model and the Fault Tree Analysis and update the FDIR analysis

In the following sections, the various categories, means of analysis, etc. will be elaborated.

Chapter 4. Fault Modeling

Fault modeling forms the core of VirSat4 FDIR and is the primary activity required to perform any FDIR analysis. Learn in this section how to use the graphical editor to perform the main analysis of VirSat4 FDIR, Fault Tree Analysis (FTA), and how to use it to build up fault models.

4.1. Modeling Fault Trees

Faults, their propagations, and inhibiting fault propagation through means of FDIR is modeled using fault trees. Fault trees are graphical models describing how faults combine with each other, propagate through the system, and eventually turn into a feared event. A fault tree is usually constructed top-down, but is read bottom-up, starting with initiating basic events. The recombination of faults is modeled via so-called gates, such as "AND" and "OR". In this section, you will learn the basics on fault tree analysis, which gates are supported by the software, and how to use the graphical editor to create a fault tree. For further in-depth information on how to perform an FTA, we refer to the standards.

4.1.1. Fault Tree Elements

The following contains a comprehensive list of fault tree elements supported by VirSat FDIR. The element descriptions are structured in the following format:

[Name of the fault tree element] **[Icon]**

— **Graphical representation**

[Representation in the graphical editor]

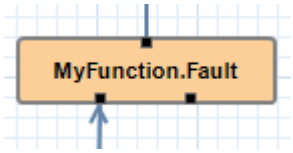
— **Description**

[Purpose of the element, its propagation behavior, additional parameters, etc.]

In addition to its inherent parameters, each fault tree node also has a name, a list of inputs, and a list of outputs. For gates, the name is by default the type of the node.

Fault 

— **Graphical representation**

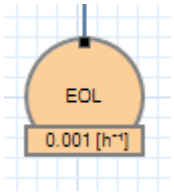


— **Description**

Faults represent logical, named events. They are used either to represent a *top-level event* of a fault tree or an *intermediate event*. Faults are also the logical containers for all other fault tree elements. As such, deleting a fault also deletes all contained elements such as gates, propagations, analysis information, etc. A fault always propagates if at least one input fails. It is, however, recommended to only have one input per fault (plus basic events).

BasicEvent

— **Graphical representation**



— **Description**

Basic events typically form the leaf elements of a fault tree. They represent basic anomalies that are not further broken down in the course of the fault tree analysis. In practice, basic events most commonly correspond to causes of equipment failure. A basic event is always directly associated to a fault. All fault propagations in a fault tree ultimately originate from basic events. Ideally, a basic event is equipped with a failure rate, quantifying its likelihood of occurrence over time. The failure rate states how often the basic event is expected to occur within a time unit. Optionally, it may also be equipped with a repair rate which conversely captures the likelihood of repair over time. Finally, a basic event may also be equipped with a cold failure rate, which comes into play when interacting together with the SPARE gate.

Propagation

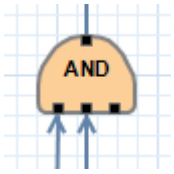
— **Graphical representation**

— **Description**

Fault propagations are the edges of a fault tree, and connect the fault tree nodes. A fault propagation has a direction. It connects the output of a fault tree node with the input of another fault tree node. Since fault trees are acyclic graphs, fault propagations may not create any cycles.

AND

— **Graphical representation**

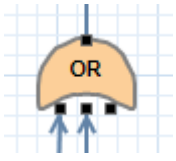


— **Description**

A gate that propagates if all inputs have failed.

OR

— **Graphical representation**

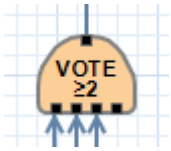


— **Description**

A gate that propagates if at least one input has failed.

VOTE

— **Graphical representation**

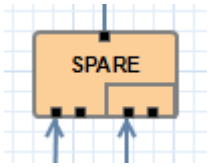


— **Description**

A gate, that is also equipped with a voting threshold k, and propagates if at least k inputs have failed. The voting threshold has to be at least 1.

SPARE

— **Graphical representation**



— **Description**

A gate with two types on inputs: Primaries and spares. If at least one primary input fails, the SPARE gate activates and claims one of the spares. Should no spares be available or failed, then the SPARE gate propagates. All spares are considered to be dormant. This means that contained basic events will use their cold failure rate, instead of their hot failure rate, as long as they are unclaimed. Once a spare is claimed, it is set to be activated and its hot failure rate is used again. Spares are claimed from left to right. In the case of a repair, the SPARE gate switches back. Spares may be shared between spare gates. However, there must not be common nodes between spares or between spares and primaries. The only exception of this rule, are functional dependency gates.

POR

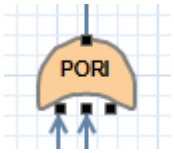
— **Graphical representation**

— **Description**

A Priority OR (POR) gate propagates if the left-most input occurs before any other input.

PORI

— **Graphical representation**

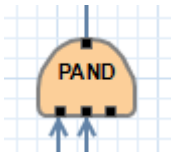


— **Description**

An Inclusive Priority OR (PORI) gate propagates if the left-most input occurs before any other input, or at the same time as another input.

PAND

— **Graphical representation**

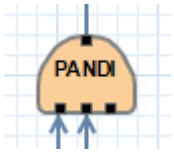


— **Description**

A Priority AND (PAND) gate propagates if the inputs fail exactly in sequence from left to right.

PANDI

— **Graphical representation**

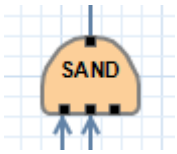


— **Description**

An Inclusive Priority AND (PANDI) gate propagates if the inputs fail exactly in sequence from left to right, or at the same time.

SAND

— **Graphical representation**

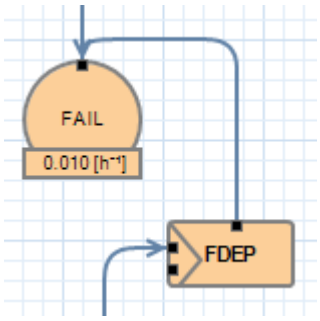


— **Description**

A Simultaneous AND (SAND) gate propagates if all inputs fail at the same time.

FDEP

— **Graphical representation**

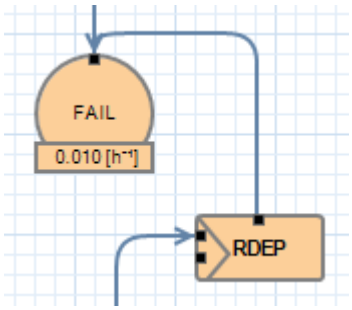


— **Description**

The functional dependency (FDEP) gate allows to trigger basic events. In the event of any input event occurring, all connected basic events get triggered.

RDEP

— Graphical representation

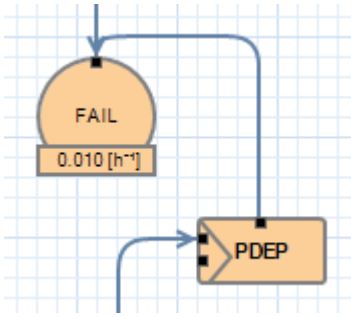


— Description

The rate dependency (RDEP) gate allows to increase the failure rate of a basic event. An RDEP is equipped with a rate change parameter r. In the event of any input event occurring, the failure rate of all connected basic events is multiplied by r.

PDEP

— Graphical representation



— Description

The probability dependency (PDEP) gate allows to trigger basic events. The PDEP gate is equipped with a trigger probability p. In the event of any input event occurring, each connected basic event is triggered with probability p.

4.1.2. The graphical Fault Tree Editor

Creating a new Fault Tree Diagram

Basic Usage

Using the Auto Layout functionality

4.2. Modeling Detection

4.3. Data exchange with the GALILEO Format

Chapter 5. Recovery Modeling

5.1. Modeling Recovery Automata

5.1.1. Recovery Automata

5.1.2. The graphical Recovery Automaton Editor

Creating a new Recovery Automaton Diagram

Basic Usage

Using the Auto Layout functionality

Chapter 6. FDIR Analysis

6.1. Configuring Analysis information

6.2. Qualitative Analysis

6.3. Quantitative Analysis

6.4. Using STORM

Chapter 7. FDIR Reporting

7.1. Using the SAVOIR/FDIR Report Template

7.2. Excel Exports

Chapter 8. FDIR Synthesis

8.1. Fault Tree Generation

8.2. Recovery Automata Synthesis

Legal - License & Copyright

Product Version:	4.10.0
Build Date Qualifier:	2020-04-23T10:01:46Z
Travis CI Job Number:	