Reliability Block Diagrams with AltaRica 3.0: part 2

A step by step introduction

The AltaRica Association

Abstract: This document shows how reliability block diagrams can be simply and efficiently assessed with AltaRica 3.0. No prior knowledge on AltaRica 3.0 is required. This document can thus be seen as a step by step introduction to AltaRica 3.0 for those who have some basic knowledge of reliability engineering modeling formalisms. It is a continuation of the previous document on reliability block diagrams.

The presentation starts with a case study description. In the second section, it shows how hierarchical reliability block diagrams can be handled with AltaRica 3.0. Finally, it presents some quantitative and qualitative assessment results.

Author(s)	Tatiana Prosvirnova, André Leblond
Reviewer(s)	Anthony Legendre
Reference	TRAS-2025-001
Date	12/12/2025

Table of Contents

1 Case study description

Consider a satellite communication system composed of a satellite **Sat**, two ground stations **GS1** and **GS2**, and two geostationary satellites **SatRelay1** and **SatRelay2** used as a relay. The satellite **Sat** communicates with the ground stations **GS1** and **GS2** via the geostationary satellites **SatRelay1** and **SatRelay2** which restransmit images to the ground stations.

The reliability block diagram representing this system is given Figure 1.

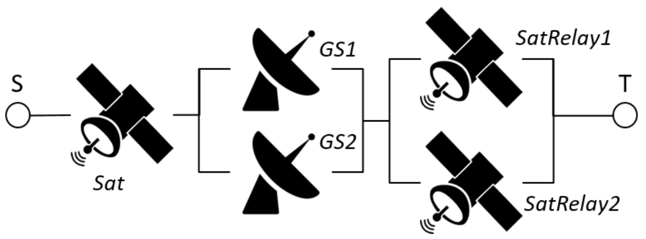

Figure 1: Reliability block diagram of the satellite communication system.

The ground station subsystem may contain antennas, transmitters, and receivers. Its reliability block diagram is represented Figure 2.

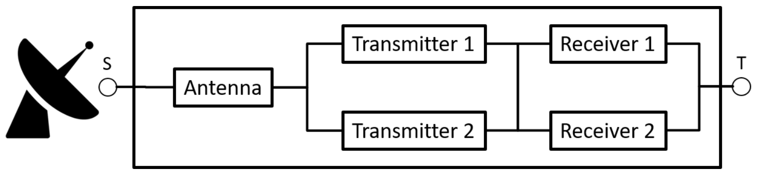

Figure 2: Reliability Block Diagram of the ground station (**GS1** or **GS2**).

The satellite subsystem is composed of batteries, transmitters and receivers. Its reliability block diagram is depicted Figure 3.

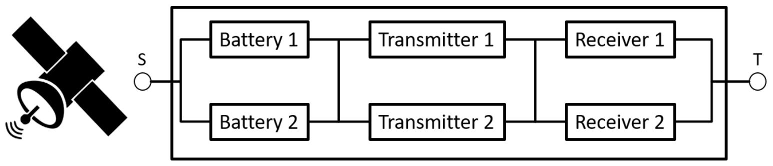

Figure 3: Reliability Block Diagram of the satellite (**Sat**, **SatRelay1** or **SatRelay2**).

The system is considered as failed when the communication between the satellite **Sat** and the ground stations **GS1** or **GS2** is lost as represented by the reliability block diagram Figure 1.

The failure rates of the components are given in Table 1.

The reliability block diagram of the Figure 1 is a hierarchical reliability block diagram because each of its blocks is represented by another reliability block diagram. In the next section, we show how hierarchical reliability block diagrams can be easily represented using structural constructs of AltaRica 3.0.

2 Hierarchical Reliability Block Diagrams

2.1 Structural constructs of AltaRica 3.0

AltaRica 3.0 offers different structural constructs to organize models into hierarchies of interconnected blocks. These constructs are stemmed from object-oriented and prototype-oriented

Table 1: Parameters

Component	**Failure rate**
Antenna	$5.0 * 10^{-5}h^{-1}$
Battery	$2.5 * 10^{-6}h^{-1}$
Receiver	$10^{-5}h^{-1}$
Transmitter	$10^{-5}h^{-1}$

programming languages (Abadi and Cardelli 1998).

The basic structural construct is a **block**, also called a **prototype**. A block is a container for variables, parameters and all the other modeling artifacts. It has a unique occurrence in the model. The simplest structuring relation is the **composition**. A block may be composed of several other blocks.

In order to be able to reuse blocks, AltaRica 3.0 introduces the notions of **class** and **instantiation** of classes. A **class** is a reusable "on-the-shelf" block, which is stored in a library and can be reused everywhere in the model via instantiation.

In some cases, it is necessary to modify or to extend a modeling unit (a class or a block) without instantiation. It can be achieved via **inheritance** relation. If a modeling unit A inherits from a modeling unit B, then A contains all the characteristics of B and adds some new characteristics.

There are cases where the same component is used in several places or to contribute to different functions of the system. In other words, a modeling unit is shared between several other modeling units. This kind of "uses" relation between modeling units is called **aggregation**.

AltaRica 3.0 offers two mechanisms to reuse models. The reuse of modeling units is done by means of instantiation of classes or via the **cloning** of blocks. If a block A is a clone of a block B, then the block A has exactly the same characteristics as the block B.

To summarize, AltaRica 3.0 offers the following constructs to organize and structure models:

– Two types of modeling units: block and class;

– Three structural relations: composition, inheritance and aggregation; and

– Two mechanisms making possible to reuse modeling elements: prototype/cloning and class/instantiation.

These structuring constructs comes from S2ML (System Structure Modelling Language). To know more about structuring constructs of modeling languages see (Batteux, Prosvirnova, and Rauzy 2018).

In the following sections, we show how hierarchical reliability block diagrams can be represented in AltaRica 3.0 using class/instance and prototype/cloning reuse mechanisms.

2.2 AltaRica 3.0 model using class/instance reuse mechanism

In this section, we show how the satellite communication system can be represented using class/instance reuse mechanism. As you can see, in the reliability block diagram Figure 1, there are two types of components:

– Satellites (Sat, SatRelay1 and SatRelay2), and

– Ground stations (GS1 and GS2).

Each component is represented itself by a reliability block diagram. To create the model of the whole system, we need to:

– Create a class representing the reliability block diagram of the satellite;

– Create a class representing the reliability block diagram of the ground station;

– Instantiate these classes as many times as shown in Figure 1;

– Create the connections between the instantiated classes;

– Define the appropriate observers in order to prepare the calculation of risk indicators.

2.2.1 AltaRica 3.0 model of the Satellite reliability block diagram

First, we create a class to represent the satellite reliability block diagram. Each component of this reliability block diagram is represented by a class `BasicBlock`, presented in the previous Step By Step (Rauzy 2024).

Figure 4 shows the AltaRica 3.0 code of this reliability block diagram. It declares a class named `SatelliteRBD` (line 1. It starts by declaring two Boolean flow variables `S` and `T` to represent the source and target nodes of the diagram (line 2).

Then the class `BasicBlock` is instantiated 6 times (lines 3- 5). Each type of components declares its own value of the failure rate `lambda` according to the values given in Table 1.

The assertion implements connections between basic blocks as shown Figure 3 (lines 7-13).

The structure of this model is the same as that presented in the previous Step by Step (Rauzy 2024).

```
1   class SatelliteRBD
2       Boolean S, T (reset = false);
3       BasicBlock Battery1, Battery2 (lambda = 2.5e-6);
4       BasicBlock Transmitter1, Transmitter2(lambda = 1.0e-5);
5       BasicBlock Receiver1, Receiver2(lambda = 1.0e-5);
6       assertion
7          Battery1.in := S;
8          Battery2.in := S;
9          Transmitter1.in := Battery1.out or Battery2.out;
10         Transmitter2.in := Battery1.out or Battery2.out;
11         Receiver1.in := Transmitter1.out or Transmitter2.out;
12         Receiver2.in := Transmitter1.out or Transmitter2.out;
13         T := Receiver1.out or Receiver2.out;
14  end
```

Figure 4: AltaRica 3.0 class for the satellite reliability block diagram

2.2.2 AltaRica 3.0 model of the ground station reliability block diagram

Second, we create a class that represents the ground station reliability block diagram. Its AltaRica 3.0 model is given Figure 5. Its structure is similar to the previous reliability block diagram.

First, the class, named `GroundStationRBD` is declared line 1.

Second, two Boolean flow variables `S` and `T`, representing the source and the target of the diagram, are declared line 2.

Third, the class `BasicBlock` is instantiated 5 times (lines 3-5).

Finally, the assertion defines the connections between basic blocks as defined in the diagram Figure 2 (lines 8-13).

```
1  class GroundStationRBD
2      Boolean S, T (reset = false);
3      BasicBlock Antenna (lambda = 5.0e-5);
4      BasicBlock Transmitter1, Transmitter2(lambda = 1.0e-5);
5      BasicBlock Receiver1, Receiver2(lambda = 1.0e-5);
6
7      assertion
8         Antenna.in := S;
9         Transmitter1.in := Antenna.out;
10        Transmitter2.in := Antenna.out;
11        Receiver1.in := Transmitter1.out or Transmitter2.out;
12        Receiver2.in := Transmitter1.out or Transmitter2.out;
13        T := Receiver1.out or Receiver2.out;
14 end
```

Figure 5: AltaRica 3.0 class for the ground station reliability block diagram

2.2.3 AltaRica 3.0 model of the satellite communication system reliability block diagram

Now, we should create the reliability block diagram of the entire system as represented in Figure 1. To do that, we need to:

- Instantiate three times the class `SatelliteRBD`;
- Instantiate twice the class `GroundStationRBD`;
- Create connections between the instances;
- Define the appropriate observers to prepare the calculation of risk indicators.

Figure 6 shows the AltaRica 3.0 model of the whole satellite communication system, represented by a block named `SatComSystem1`. It starts by declaring two Boolean flow variables `S` and `T` to represent the source and target nodes (line 2).

Then it instantiates three times the class `SatelliteRBD` (line 3) and twice the class `GroundStationRBD` (line 4).

Connections between the components are represented in the assertion part (lines 6-12).

Finally, a Boolean observer `failed` is defined line 13. An **observer** is used to declare quantities to be measured by the assessment tools. In our example it is the loss of communication between the satellite **Sat** and the ground stations **GS1** or **GS2**, which corresponds to the absence of flow in the target node `T`.

AltaRica 3.0 models given Figures 4, 5, 6 together with the model of the class `BasicBlock`, presented in the previous Step by Step (Rauzy 2024), provide the model of the whole satellite communication system.

As you can see, the class/instance reuse mechanism of AltaRica 3.0 allows an efficient modeling of hierarchical reliability block diagrams. Indeed, each reliability block diagram is represented by a class, and is reused by instantiation as many times as needed in the model.

```
block SatComSystem1
    Boolean S, T (reset = false);
    SatelliteRBD Sat, SatRelay1, SatRelay2;
    GroundStationRBD GS1, GS2;
    assertion
       S := true;
       Sat.in := S;
       GS1.in := Sat.T;
       GS2.in := Sat.T;
       SatRelay1.S := GS1.T or GS2.T;
       SatRelay2.S := GS1.T or GS2.T;
       T := SatRelay1.T or SatRelay2.T;
   observer Boolean failed = not T;
end
```

Figure 6: AltaRica 3.0 model of the satellite communication system reliability block diagram using class/instance reuse mechanism

2.3 AltaRica 3.0 model using prototype/cloning reuse mechanism

In this section we show how the satellite communication system can be represented using prototype/clone reuse mechanism. To create the model of the whole system, we need to:

- Create a block representing the reliability block diagram of the whole system;
- Inside this block
    - Create a block representing the reliability block diagram of the satellite;
    - Create a block representing the reliability block diagram of the ground station;
    - Clone the satellite and ground station blocks as many times as shown in Figure 1;
    - Create the connections between the satellite, the ground station blocks and their clones;
    - Define the appropriate observers in order to prepare the calculation of risk indicators.

The AltaRica 3.0 model of the whole satellite communication system is given Figure 7. First, we create a block to represent the reliability block diagram of the whole satellite communication system, named `SatComSystem2` (lines 1 - 42). Inside this block, we create a block, named `Sat`, to represent the reliability block diagram of the satellite (lines 3-16). Then the class `BasicBlock`, defined in the previous Step by Step (Rauzy 2024), is instantiated 6 times (lines 4 - 6). Flow variables S and T are defined to represent the source and the target nodes of the block `Sat` (line 7). The assertion implements connections between basic blocks as shown on Figure 7 (lines 9 -15).

Second, the block `Sat` is cloned twice as `SatRelay1` and `SatRelay2` (lines 17 - 18.

Third, we create a block representing the reliability block diagram of the ground station, named `GS1` (lines 19 - 42). Its AltaRica 3.0 model is also given in Figure 7. Its structure is similar to the reliability block diagram of the block `Sat`.

Finally, block GS1 is cloned once as GS2 (line 32). Connections between the blocks and the clones are represented in the assertion part (lines 35 - 40).

A Boolean observer `failed` is defined in line 41.

The AltaRica 3.0 model given Figure 7 together with the AltaRica 3.0 model of basic blocks given in the previous Step by Step represent the hierarchical reliability block diagram of the

```
block SatComSystem2
    Boolean S, T (reset = false);
    block Sat
      BasicBlock Battery1, Battery2 (lambda = 2.5e-6);
      BasicBlock Transmitter1, Transmitter2(lambda = 1.0e-5);
      BasicBlock Receiver1, Receiver2(lambda = 1.0e-5);
      Boolean S, T ( reset = false );
      assertion
        Battery1.in := S;
        Battery2.in := S;
        Transmitter1.in := Battery1.out or Battery2.out;
        Transmitter2.in := Transmitter1.in;
        Receiver1.in := Transmitter1.out or Transmitter2.out;
        Receiver2.in := Transmitter1.out or Transmitter2.out;
        T := Receiver1.out or Receiver2.out;
    end
    clones Sat as SatRelay1;
    clones Sat as SatRelay2;
    block GS1
     BasicBlock Antenna (lambda = 5.0e-5);
     BasicBlock Transmitter1, Transmitter2 (lambda = 1.0e-5);
     BasicBlock Receiver1, Receiver2 (lambda = 1.0e-5);
     Boolean S, T ( reset = false );
     assertion
        Antenna.in := S;
        Transmitter1.in := Antenna.out;
        Transmitter2.in := Antenna.out;
        Receiver1.in := Transmitter1.out or Transmitter2.out;
        Receiver2.in := Receiver1.in;
        T := Receiver1.out or Receiver2.out;
    end
    clones GS1 as GS2;
    assertion
       S := true;
       Sat.S := S;
       GS1.S := Sat.T;
       GS2.S := Sat.T;
       SatRelay1.S := GS1.T or GS2.T;
       SatRelay2.S := GS1.T or GS2.T;
       T := SatRelay1.T or SatRelay2.T;
   observer Boolean failed = not T;
end
```

Figure 7: AltaRica 3.0 model for the satellite communication system reliability block diagram

satellite communication system. It uses prototype/cloning reuse mechanism to represent hierarchical reliability block diagrams. It is semantically equivalent to the model defined in the previous section.

3 Model assessment

Both models designed in the previous section are equivalent and can be assessed by compiling them into Boolean equations (fault trees). The compiler to Boolean equations (fault trees) produces a model at Open-PSA format (Epstein and Rauzy 2008) so that it can be assessed with the XFTA calculation engine (Rauzy 2020). XFTA produces the minimal cutsets given in Table 2.

Table 2: Minimal cutsets calculated from the generated Boolean equations (fault tree).

Order	MCS		
2	GS1.Antenna.failure	GS2.Antenna.failure	
2	Sat.Receiver1.failure	Sat.Receiver2.failure	
2	Sat.Battery1.failure	Sat.Battery2.failure	
2	Sat.Transmitter1.failure	Sat.Transmitter2.failure	
3	GS1.Transmitter1.failure	GS1.Transmitter2.failure	GS2.Antenna.failure
3	GS1.Antenna.failure	GS2.Transmitter1.failure	GS2.Transmitter2.failure
3	GS1.Antenna.failure	GS2.Receiver1.failure	GS2.Receiver2.failure
3	GS1.Receiver1.failure	GS1.Receiver2.failure	GS2.Antenna.failure
4	SatRelay1.Receiver1.failure	SatRelay1.Receiver2.failure	
	SatRelay2.Receiver2.failure	SatRelay2.Receiver1.failure	
4	SatRelay1.Receiver1.failure	SatRelay1.Receiver2.failure	
	SatRelay2.Battery2.failure	SatRelay2.Battery1.failure	
4	SatRelay1.Receiver1.failure	SatRelay1.Receiver2.failure	
	SatRelay2.Transmitter2.failure	SatRelay2.Transmitter1.failure	
4	SatRelay1.Battery1.failure	SatRelay1.Battery2.failure	
	SatRelay2.Receiver2.failure	SatRelay2.Receiver1.failure	
4	SatRelay1.Battery1.failure	SatRelay1.Battery2.failure	
	SatRelay2.Battery2.failure	SatRelay2.Battery1.failure	
4	SatRelay1.Battery1.failure	SatRelay1.Battery2.failure	
	SatRelay2.Transmitter2.failure	SatRelay2.Transmitter1.failure	
4	GS1.Transmitter1.failure	GS1.Transmitter2.failure	
	GS2.Transmitter2.failure	GS2.Transmitter1.failure	
4	GS1.Transmitter1.failure	GS1.Transmitter2.failure	
	GS2.Receiver2.failure	GS2.Receiver1.failure	
4	GS1.Receiver1.failure	GS1.Receiver2.failure	
	GS2.Transmitter2.failure	GS2.Transmitter1.failure	
4	GS1.Receiver1.failure	GS1.Receiver2.failure	
	GS2.Receiver2.failure	GS2.Receiver1.failure	
4	SatRelay1.Transmitter1.failure	SatRelay1.Transmitter2.failure	
	SatRelay2.Receiver2.failure	SatRelay2.Receiver1.failure	
4	SatRelay1.Transmitter1.failure	SatRelay1.Transmitter2.failure	
	SatRelay2.Battery2.failure	SatRelay2.Battery1.failure	
4	SatRelay1.Transmitter1.failure	SatRelay1.Transmitter2.failure	
	SatRelay2.Transmitter2.failure	SatRelay2.Transmitter1.failure	

XFTA also allows to calculate probabilistic indicators related to the top event or the intermediate events of the fault tree. Figure 8 shows the evolution of the unreliability of the studied system from 0 to a given mission time T, here $T = 8760h$.

Assuming the system's unreliability must stay below 0.1 over a year, the results show that this target has not been met.

After analyzing the minimal cut sets (MCS), we see that the components that contribute most to the system's unreliability are the the antennas of the ground stations $GS1$ and $GS2$ and the satellite Sat components.. To enhance overall reliability, we could introduce redundancy for the ground station antennas and consider using higher-reliability components for the satellite.

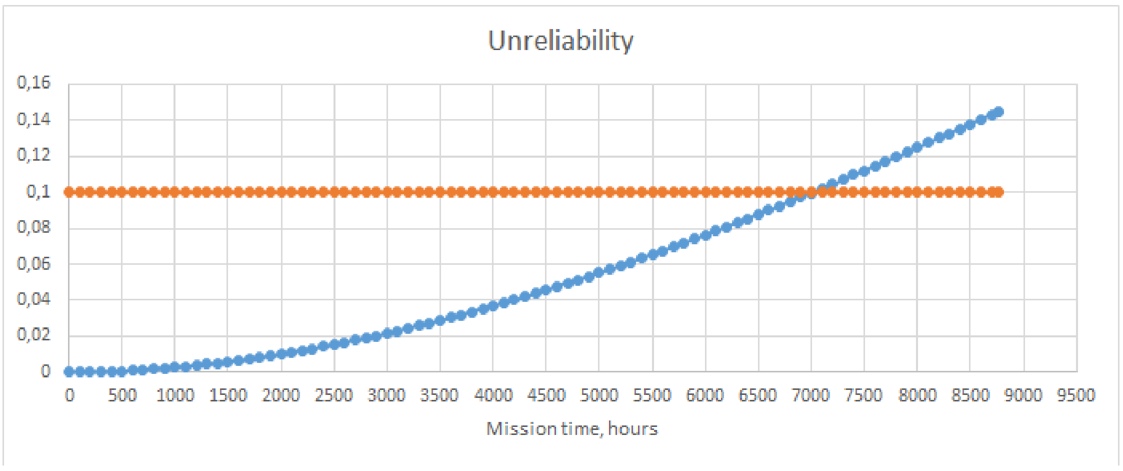

Figure 8: Unreliability calculated from the generated Boolean equations (fault tree).

4 References

Abadi, Mauricio and Luca Cardelli (1998). *A Theory of Objects*. New-York, USA: Springer-Verlag. ISBN: 978-0387947754 (cited on page 5).

Batteux, Michel, Tatiana Prosvirnova, and Antoine Rauzy (2018). "From Models of Structures to Structures of Models". In: *2018 IEEE International Systems Engineering Symposium (ISSE) (IEEE ISSE 2018)*. Rome, Italy (cited on page 5).

Epstein, Steven and Antoine Rauzy (2008). *Open-PSA Model Exchange format, version 2.0d*. available at www.altarica-association.org (cited on page 10).

Rauzy, Antoine (2020). *Probabilistic Safety Analysis with XFTA*. Les Essarts le Roi, France: AltaRica Association. ISBN: 978-82-692273-0-7 (cited on page 10).

— (2024). *Reliability Block Diagrams with AltaRica 3.0: part 1*. Technical Reports of the AltaRica Association TRAS-2024-001. Les Essarts le Roi, France: AltaRica Association (cited on pages 6–8).

A Authoring Models with AltaRica 3.0 Workshop

AltaRica 3.0 Workshop is an integrated modeling environment dedicated to probabilistic risk and safety analyses of complex technical systems. It supports two modeling languages: AltaRica 3.0 and S2ML+SBE. Both languages belong to the S2ML+X family.

AltaRica 3.0 Workshop contains:

- AltaRica Wizard, a graphical interface for authoring AltaRica 3.0 and S2ML + SBE models.
- A complete set of assessment tools:
  - An interactive simulator;
  - A generator of stochastic Boolean equations;
  - A stochastic simulator;
  - A generator of critical sequences;
  - A calculation engine for stochastic Boolean equations XFTA.

This section explains how to author models with AltaRica Wizard.

A.1 Projects

AltaRica Wizard manages models through projects. A *project* contains one or more files organized into folders. Files of a project can be AltaRica source files (having usually the extension "`.alt`"), as well as other types of files generated by assessment tools.

Projects can be created, populated with folders and files, saved and (re)loaded from AltaRica Wizard. They are described into XML files with the extension "`.ar3w`". Project description files can thus be edited manually, although this should be done with care.

Files and folders of a project are those of the underlying operating system. In particular, the project itself is located into a folder of the underlying operating system, the *project folder*. All paths to folders and files are considered relatively to the project folder and the file describing the project is saved into this folder. This is the reason why it is highly recommended to organize all of the files of a project under the project folder. In this way, the project can be moved from one place to another one just by moving the project folder. The same principle applies indeed to project copy: to duplicate a project it suffices to copy the folder that contains this project.

Folders are automatically added to and removed from projects when adding and removing files. This means that a project contains a folder if and only if this folder contains a file of the project (possibly located in a sub-folder, a sub-sub-folder, . . . of that folder). *A contrario*, sub-folders of the project folder do not belong automatically to the project. Only those containing a file of the project do.

Creating a new project, opening an existing one and saving the current project is done via the menu "File". Creating a new project implies selecting a folder for that project. By default, AltaRica Wizard creates this folder for you. Nevertheless, it is possible to create a project without creating a new folder.

A.2 Source Files

Once the project created AltaRica source files can be added to this project. AltaRica source files must have the extension ".alt".

It is possible to create AltaRica source files directly via the menu “Project”. Another solution consists in creating AltaRica source files by means of an external text editor and then to add them to the project, also via the menu “Project”.

When tools are launched, all AltaRica source files of the project are gathered. The model is thus made of all domains, blocks and classes declared in AltaRica source files.

The AltaRica step by step series

AltaRica 3.0 is an object-oriented modeling language dedicated to performance analyses of complex technical systems. It makes it possible to design highly reusable stochastic discrete event models. The AltaRica step by step series aims at showing how different types of systems can be represented in AltaRica 3.0 and how their performance—to be taken in a broad sense—can be assessed by means of AltaRica 3.0 assessment tools.

Each document of the series is a step by step introduction to AltaRica 3.0. We tried to minimize the number of pre-requisites so that all engineers or students in engineering disciplines can take benefit of these presentations.

Beyond being a simple collection of smooth introductions to AltaRica 3.0, the AltaRica step by step series aims at gathering a versatile set of reusable modeling patterns. Using these patterns makes the learning curve steeper and the modeling process more efficient.

AltaRica 3.0 results from two decades of active academic research and industrial experience. It comes with AltaRica Wizard, an integrated modeling and simulation environment. AltaRica Wizard is coupled with a complete set of assessment tools enabling a wide variety of analyses. The specification of the language and the integrated modeling and simulation environment are developed by the not for profit AltaRica Association. They can be used free of charge for research and education purposes. The AltaRica project is supported by several academic institutions and world leading industrial partners.

Already published in AltaRica step by step series

TRAS-2024-001: Reliability Block Diagrams with AltaRica 3.0: part 1

TRAS-2025-001: Reliability Block Diagrams with AltaRica 3.0: part 2