information

MDPI

Article

Modelling and Resolution of Dynamic Reliability Problems by the Coupling of Simulink and the Stochastic Hybrid Fault Tree Object Oriented (SHyFTOO) Library

Ferdinando Chiacchio [1,*], Jose Ignacio Aizpurua [2], Lucio Compagno [1], Soheyl Moheb Khodayee [3] and Diego D'Urso [1]

[1] Department of Electrical, Electronic and Computer Engineering, University of Catania, 95131 Catania, Italy
[2] Department of Electronics & Computer Science, Mondragon University, 20500 Mondragón, Spain
[3] School of Industrial engineering, Islamic Azad University, South Tehran Branch, Tehran 1584743311, Iran
[*] Correspondence: chiacchio@dmi.unict.it; Tel.: +39-0957382412

Received: 5 August 2019; Accepted: 7 September 2019; Published: 11 September 2019

Abstract: Dependability assessment is one of the most important activities for the analysis of complex systems. Classical analysis techniques of safety, risk, and dependability, like Fault Tree Analysis or Reliability Block Diagrams, are easy to implement, but they estimate inaccurate dependability results due to their simplified hypotheses that assume the components' malfunctions to be independent from each other and from the system working conditions. Recent contributions within the umbrella of Dynamic Probabilistic Risk Assessment have shown the potential to improve the accuracy of classical dependability analysis methods. Among them, Stochastic Hybrid Fault Tree Automaton (SHyFTA) is a promising methodology because it can combine a Dynamic Fault Tree model with the physics-based deterministic model of a system process, and it can generate dependability metrics along with performance indicators of the physical variables. This paper presents the Stochastic Hybrid Fault Tree Object Oriented (SHyFTOO), a Matlab® software library for the modelling and the resolution of a SHyFTA model. One of the novel features discussed in this contribution is the ease of coupling with a Matlab® Simulink model that facilitates the design of complex system dynamics. To demonstrate the utilization of this software library and the augmented capability of generating further dependability indicators, three different case studies are discussed and solved with a thorough description for the implementation of the corresponding SHyFTA models.

Keywords: dependability; monte carlo simulation; multi-state systems; repairable dynamic fault trees; dynamic systems; hybrid simulation

1. Introduction

Dependability is the ability to avoid service failures that are more frequent and severe than is acceptable, and it is comprised of the following attributes [1]:

- reliability: continuity of correct service;
- availability: readiness for correct service;
- safety: absence of catastrophic consequences on the user(s) and environment;
- maintainability: ability to undergo modifications and repairs.

The dependability assessment of a system is a crucial activity for engineers and industrial stakeholders so as to ensure the dependable operation of the system. Several different stochastic formalisms have been developed to evaluate the different dependability attributes of complex and dynamic systems. Among these, there are well-known classical methods such as Reliability Block Diagrams (RBD), Fault Tree Analysis (FTA), Continuous Time and Discrete Markov Chains (CTMC/DMC), Stochastic Petri Nets (SPN), Bayesian Networks (BN), and several derived methods that aim at improving their limitations (e.g., Fuzzy FTA, Boolean Driven Markov Process, Dynamic Fault Tree, Reliability Block Diagrams, etc.). Each of these formalisms has different advantages and limitations [2], and accordingly, the choice of the appropriate dependability assessment methodology depends on the complexity of the system, amount of available information, and required key performance indicators. One of the main shortcomings in the above-mentioned formalisms is that they cannot model the dependency between the failure behavior of the components and the system working conditions. In fact, the probability distribution of the time to failure of the system components is fixed and must be provided as a static input of the model.

Driven by the need for an accurate dependability analysis process, in order to tackle the previous limitation, during the last few decades, researchers have developed hybrid models that are able to account for the stochastic and deterministic failure and operation processes including random phenomena and physics-based evolutions. The branch of dependability modelling that considers stochastic and deterministic processes is known as Dynamic Probabilistic Risk Assessment (DPRA) or Dynamic Reliability [3]. In a DPRA model, the physics of a process can influence the stochastic behavior of the system components. One of the main difficulties of such a modelling process is the level of detail that influences the model size and its complexity [4].

The system dependability assessment is carried out by a team of experts that may eventually include risk practitioners, mathematicians, and engineers [5]. The common goal is the conception of a model that defines the system failure and operation processes to quantify the occurrence probability of undesired events. This process requires a detailed knowledge of the system processes and the ability to describe them with a stochastic model. Computer-aided tools can ease this activity, and the literature presents several contributions that demonstrate the potential of academic tools and their main scope of application with respect to the process under analysis and the stochastic modelling approach [6–22].

Following this research stream, in this paper, the authors focus on the modelling of Stochastic Hybrid Fault Tree Automaton (SHyFTA) and illustrate how to use the SHyFTOO Matlab® toolbox to implement and solve general DPRA problems. SHyFTA is a stochastic formalism of DPRA based on Repairable Dynamic Fault Tree (RDFT) [21], and SHyFTOO is a software library presented in a previous research paper by the authors [22]. The breakthrough of the framework and of the proposed tool, compared to the competitive tools that are now available in the industrial scenario, stands in its native hybrid feature, which is able to combine Time-Driven and Discrete-Event simulation paradigms to solve the continuous dynamic of a physical process and to evaluate at any relevant time event the reliability status of the system.

The aim of this contribution is twofold:

(i) illustrate the integration of a SHyFTA model coded with the SHyFTOO library with Matlab® Simulink toolbox;
(ii) present new emerging capabilities of a simulated SHyFTA model that, thanks to the coupling of these two computer-aided components, allows the analysis of more complex dependencies and the measurement of several key performance indexes of a system process.

The rest of the paper is organized as follows. Section 2 resumes the evolution of the stochastic Fault Tree Analysis (FTA) toward the hybrid version that integrates stochastic and deterministic processes. Section 3 illustrates how to model a SHyFTA model with the SHyFTOO library and Simulink with the aid of a case study. Section 4 presents the resolution of three different models to illustrate a portfolio of applications that can be analysed, exploiting the SHyFTOO library to compute the main dependability attributes and other key performance indicators. In Section 5, conclusions and future works are summarized. Moreover, the Appendices A and B contain several information about the SHyFTOO library that can aid interested readers and practitioners in using this tool.

2. The Evolution of Fault Tree Analysis

Figure 1 shows a breakdown of the three main categories of the stochastic methods used in the dependability assessment, and Table 1 displays their characteristics and modelling hypotheses. As shown in Figure 1, Fault Tree Analysis (FTA) is a common pattern of each category because it has been object of continuous modification over the last few years, improving its expressiveness and accuracy.

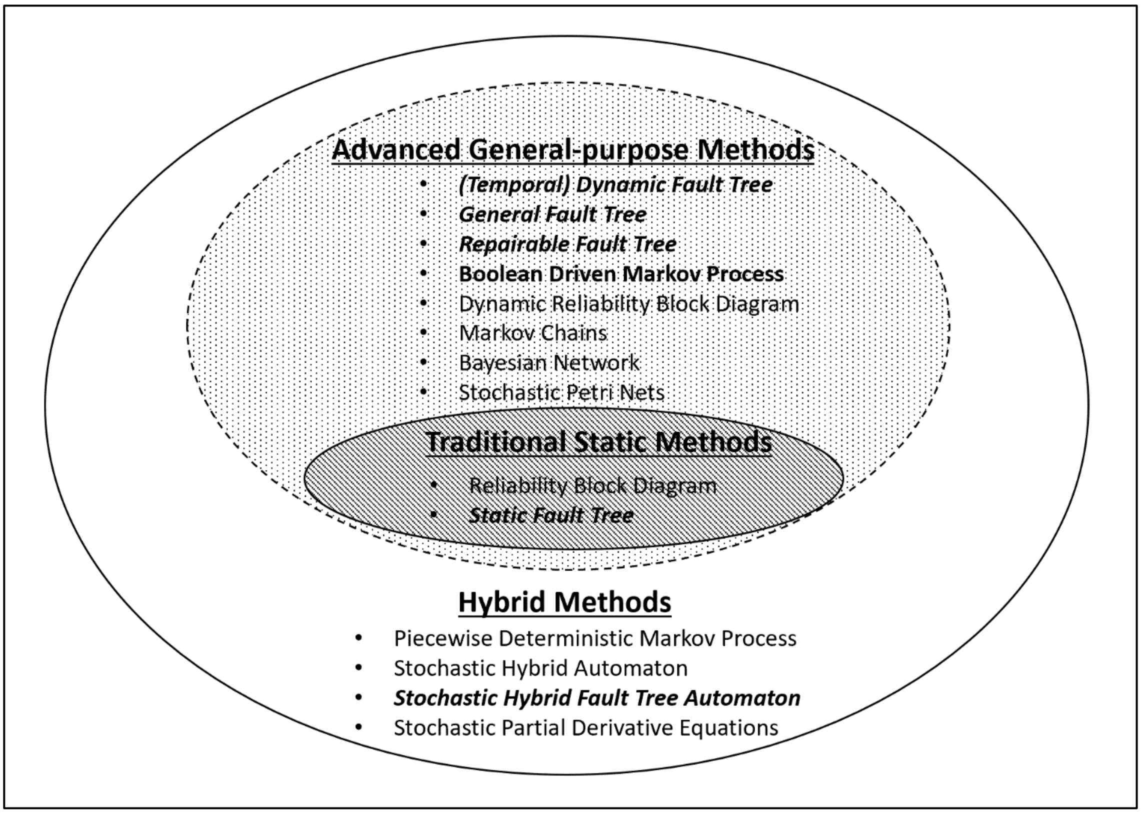

Advanced General-purpose Methods
(Temporal) Dynamic Fault Tree
General Fault Tree
Repairable Fault Tree
Boolean Driven Markov Process
Dynamic Reliability Block Diagram
Markov Chains
Bayesian Network
Stochastic Petri Nets
Traditional Static Methods
Reliability Block Diagram
Static Fault Tree
Hybrid Methods
Piecewise Deterministic Markov Process
Stochastic Hybrid Automaton
Stochastic Hybrid Fault Tree Automaton
Stochastic Partial Derivative Equations

Figure 1. Breakdown of the categories of stochastic modelling methods for the dependability assessment.

Table 1. Main characteristics of the stochastic models used in dependability assessment.

Process	**Dependability**		**Dynamic Reliability**
	▨ **Traditional Models**	░ **Advanced & GP Models**	□ **Hybrid-Dynamic Models**
Physical	Static working conditions; Single-state operating components	Static working conditions;	Dynamic working conditions; Multi-state operating components
Stochastic	Bi-State (working or failed) components; Fixed probability of failure; Independence of components	Multi-state degrading components; Fixed probability of failure; Time-event sequence dependencies	Multi-state degrading components; Dynamic probability of failure; Time-event sequence dependencies

Static Fault Tree (SFT) methodology was conceived in 1962 at Bell Laboratories by H.A. Watson, under a US Air Force Ballistics Systems Division contract to evaluate the Minuteman I Intercontinental Ballistic Missile (ICBM) Launch Control System [23]. The SFT formalism provides an easy way to model and understand the failure behaviour of a system. However, its underlying assumptions limit the application to systems with working and failed states. The construction of a fault tree follows a top-down approach that starts with the identification of a major undesired event (top event) as consequence of a cascade of intermediate events which can be modelled using the Boolean logic gates AND and OR. The leaves of the fault tree are called "basic events" and cannot be further decomposed.

They are characterized by the probability distribution of the time to occur (or to fail) of the system components that are generally provided by the components manufacturer. Static Fault Tree belongs to the class of Boolean methods. Therefore, they can be solved with analytic algorithms that provide an exact result [24–29] also in the case of non-coherent systems, described with negative logic gates [30]. Static Fault Tree is currently one of the preferred methodologies in the industrial sector [31,32] and, in authors' opinion, this prevalence will be maintained as long as the "Advanced & General-Purpose" and the "Hybrid" methodologies do not become of easy access and understanding for the industrial risk practitioners.

A state-of-the-art survey demonstrates that several modifications of the original FTA formalism have been attempted to overcome the limitations of the SFT. These techniques are grouped into the second category of models, here named as "Advanced & General-Purpose Methods". The main improvements over the "Traditional Static Methods" focus on the capability to model complex failure dependencies such as time-event sequences, multi-state degradation, and standby policies [33]. However, various methodologies do not have the same properties, and the use of one or the other approach depends on the dependability problem under study.

Among the various techniques, Dynamic Fault Tree (DFT) [34] is probably the most known method which addresses temporal events through dynamic gates. DFTs kick-started the proliferation of other formalisms that has continued for two decades now. For instance, a recent research paper [35] proposed a new fault tree modeling paradigm named Temporal Dynamic Fault Trees (TDFT) that is able to calculate the reliability and availability of a system exploiting the notion of soft-faults, i.e., temporary events that disappear after the source of the interference is no longer present. Other contributions [36,37] integrated the primitives of parametric, Repairable Dynamic Fault Trees (RDFT), and introduced a new formalism called Generalized Fault Trees (GFT) with the aim of reducing the cost of analysis and enabling the evaluation of different repair policies. Moreover, they generalized this methodology to handle parametric fault trees and dynamic fault trees. Authors in [38] suggested Boolean Logic Driven Markov Process (BDMP) as an equivalent formalism to evaluate DFT characterized by repair transitions. In [39], an ad-hoc approach for dealing with RDFTs is proposed. The main novelty is the introduction of a simplifying assumption of independency among gate inputs. In RDFTs, the occurrence of a basic event needs to be considered along with its repair process. There are system state-transitions, and these are handled using the concept of renewal processes instead of the commonly used Markov chains.

Other suggested improvements can be found in the literature, such as [40], which integrated prognostic estimations into dependability assessment to avoid using average failure rate data, which leads to outdated system-level dependability estimation, and proposed a new framework, which results in more realistic and accurate dependability predictions. A methodology for combining the Fuzzy logic and FTA are discussed in [41] in order to cope with the information uncertainty due to limited data of the failure components probability, whereas [42] presents an improved methodology for analysing the qualitative importance of components in the functional and reliability structures of a system.

The resolution of the above-mentioned models is not as simple as SFT, and accordingly, the following three techniques are generally adopted: (i) algebraic methods, (ii) conversion into an equivalent model, and (iii) simulation. As far as it concerns the algebraic methods, [43,44] presented the algebraic relations describing the temporal priorities and the quantitative analysis calculation of the gates. Similar works are addressed in [45], which presents a special version of sequence algebras, representing failures of non-repairable components, and proposes a "sequence decision diagram" to encode sets of data.

Under some hypotheses, a fault tree can be converted into its equivalent state-space model like Markov chains or Stochastic Petri Nets. However, this approach has certain limitations. One is that the transformation can be made only if the model meets the needed requirements in the secondary modelling method, e.g., only components with exponential failures can be modeled using Markov chains. The other is the exponential state-space growth of the secondary models with a large number of components. Nevertheless, in a recent contribution [46], a simplified hierarchical approach shows that it is possible to convert a DFT by the mean of a limited state space constituted by five statuses and solve it through the Semi-Markov Process theory. In this way, the exclusive use of the exponential probability distributions is overcome. Unfortunately, the same approach is not much effective for RDFT.

Finally, the simulation approach based on Monte Carlo simulation (MCS) offers a modeling flexibility that overcomes the limitations of the previous methods. Many repeated iterations of the system evolutions are generated to achieve an acceptable accuracy in estimations. There are practical case studies that have used MCS to solve complex systems modeled by a fault tree. In [47], dynamic gates with repairs have been used to model an electrical power supply system of a nuclear power plant (NPP), and due to its complexity, MCS is used to solve this model. The main issues of the simulation are the computational speed and approximation of the results that can be unacceptable for real-time applications that require a fast response. Fortunately, this is not the case of offline dependability assessments whose results, generally, are included in the safety and risk report documentation of the enterprise. Recent contributions [48] propose relieving solutions for alleviating this issue and tackle rare events.

Although the category of "Advanced & General-Purpose Methods" is powerful, the main limitation of all these formalisms is that they are purely stochastic models. With these models, the boundary conditions must be fixed as input of the model, whereby the description of the stochastic events of the model gets limited to the usage of static probability density functions. Hybrid methods [49–52], falling under the umbrella of Dynamic Reliability, alleviate this issue, and, as shown in Figure 1, they embrace the scope covered by the other categories. Dynamic Reliability aims to relax the rigid hypotheses of traditional reliability, enabling the possibility to model multi-state systems and consider changes of the nominal design condition of a system. It can consider numerous characteristics of complex systems, such as inclusion of environmental dependencies, interactions between continuous process variables and system components, and stochastic and deterministic behaviours evolving in time. As matter of fact, a component does not always operate around the nominal design operative conditions, resulting in deviations of performance and wearing-out. The formulation and resolution of a Dynamic Reliability problem is not an easy task because, as shown in Equation (1), the mathematical relationship to write and solve it has to consider the stochastic events affecting the dynamic evolution of the working and operational conditions W (Equation (2)) and the variation of the physical dynamic behavior (Equation (3)), by means of a set of Partial Differential Algebraic Equations (PDAE).

$$R(t) = \mathbb{P}(S(L) \geq T_M,\ W,\ X(t)) \tag{1}$$

where $\mathbb{P}$ is the probability that the time to failure of the system (S) is greater than the mission time T_M and L is the aging variable. The working and operating conditions W can be described by the mean of a non-linear functional Γ that depends on time t, on the system dynamics X(t) and on a vector Λ of stochastic variables.

$$W = \Gamma(t,\ X(t), \Lambda) \tag{2}$$

Finally, the general formulation of the PDAE problem, in terms of thermo-fluid, chemical, and rotational dynamic laws (e.g., mass energy balance, heat transfer equations) has the following form:

$$\begin{cases} \frac{dX(t)}{dt} = F\left(X(t),\ \frac{\partial X}{\partial \zeta_i}(t),\ y(t),\ u(t),\ t\right) \\ B\left(X(t),\ \frac{\partial X}{\partial \zeta_i}(t),\ y(t),\ u(t),\ t\right) = 0 \end{cases} \tag{3}$$

where t denotes the time, y and u are the vectors of differential variables, algebraic variables and inputs, respectively. The vectors X and y are, respectively, the system and the process variables (state variables). F and B are vector functions. The partial derivative base vector, ζ, is an appropriate distribution domain, usually expressing geometry dimensions (e.g., length, width, radius) [53].

In order to simplify the modelling of such problems, recent studies have proposed methodologies based on the separation of concerns that allows the independent modelling of the deterministic and stochastic processes [54]. In this category of methods, the Stochastic Hybrid Fault Tree Automaton (SHyFTA) is the one that mostly resembles to the FTA [55]. A SHyFTA model is comprised of two interdependent models. The deterministic model and the stochastic model with the latter taking the form of a Repairable Dynamic Fault Tree. The formal definition states that a SHyFTA is a 13-uplet.

$$\mathrm{SHyFTA} = \{S,\ \mathcal{E},\ \mathrm{X},\ \mathrm{Y}, \delta,\ H, G,\ F,\ P,\ GA, BE, T,\ C\}$$

- S is a finite set of discrete states $\{S_{\mathrm{D}}, S_{\mathrm{S}}\}$, where S_{D} is the subset of deterministic states and S_{S} of the stochastic one;
- $\mathcal{E}$ is a finite set of events $\{\mathcal{E}_{\mathrm{D}},\mathcal{E}_{\mathrm{S}}\}$, where $\mathcal{E}_{\mathrm{D}}$ is the subset of deterministic events and $\mathcal{E}_{\mathrm{S}}$ of the stochastic one;
- X is a finite set of real variables evolving in time $\{\mathrm{x}_1, \ldots, \mathrm{x}_{\mathrm{n}}\}$;
- Y is a finite set of arcs of the form (s, ε_{j}, G_{k}, s') where s and s' are, respectively, the origin and the goal discrete states of the arc k, ε_{j} is the event associated with this arc, G_{k} is the guard condition on X in state s;
- G is a finite set of guard condition functions on each real variable X_{i} on the state $s_{\mathrm{j.}}$
- δ: $S \times \mathrm{X} \to (\mathbb{R}^{n+} \to \mathbb{R})$ is a function of "activities", describing the evolution of real variables in each discrete state;
- H is a finite set of clocks on $\mathbb{R}$ that identify the firing of a deterministic or a stochastic event;
- F: $H \times S \times \mathrm{X} \to (\mathbb{R}^{n+} \to [0,1])$ is an application that associates a distribution function to the stochastic events $\mathcal{E}$S, according to the clock H, the system evolution X and the discrete state S;
- P is the instantaneous probability to be in $s_{\mathrm{i}} \in S_{\mathrm{S}}$;
- GA is the finite set of gates of the fault tree model;
- BE is the finite set of basic events of the fault tree model. The set BE contains a subset of novel conception, the Hybrid Basic Events (HBE), $\mathrm{HBE} \subseteq \mathrm{BE} : \forall\ be_i \in \mathrm{HBE}\, \exists\, \mathrm{f}_i : H \times S \times \mathrm{X} \to (\mathbb{R}^{n+} \to [0,\ 1])$. This last characterization states that HBEs are those set of basic events whose failure distribution depends on the evolution of the system and vary with continuity in time;
- T is the top event of the fault tree corresponding with the output of the main gate;
- C is the set of connections between gates and basic events.

For a thorough description of the SHyFTA formalism, readers can refer to [55].

The Stochastic Hybrid Automaton is not easy to use and solve because it models, without constraints, any type of dependency. For this reason, the most appropriate resolution method is based on Monte Carlo simulation. In order to favour the design of a model, an ad-hoc software library, named SHyFTOO, has been specifically developed under the Matlab® framework [22]. SHyFTOO is characterized by a computer algorithm that combines Time-Driven and Discrete-Event simulation [22,55,56]. The main disadvantage of this library is that it can require numerous iterations and long runs to retrieve accurate results. However, the authors are confident that the speed of penetration of new paradigms based on parallel and cloud computing [57] can alleviate this issue. Moreover, the deepness of a model [4], meant as the capability to describe the accuracy of a phenomena within a correct sizing of the boundary conditions (and to avoid too complex models), must be considered in order to design a model that is effective enough for the type of analysis to achieve. In fact, while traditional models are based on fixed probability density functions, in the Dynamic Reliability, the specific experience and knowledge of the system equipment and of the processes by the modelers is an important factor that affects the accuracy of the final model (e.g., the relationships between components-failure dynamics and the system working conditions). The proposed version of SHyFTOO has been coded in Matlab®, which offers a high-level user-friendly interface and facilitates the modelling of dynamic systems.

In the Appendix A, all the main components of the SHyFTOO library are described. Moreover, the novel standard template Simulink blocks that belong to the SHyFTOO library package and that allow a DFT to couple with a Simulink dynamic system of Matlab® are presented.

3. Case Study

This section describes the main case study of this manuscript and the SHyFTA model implemented with the SHyFTOO library. Its resolution is then presented in Section 4 together with other case study results. To better understand the process under analysis, Figure 2 shows a representation of the SHyFTA model hereby described. As it will be shown, it is implemented by using several built-in Simulink blocks that are combined with the standard template blocks of the SHyFTOO library discussed in the Appendix A.

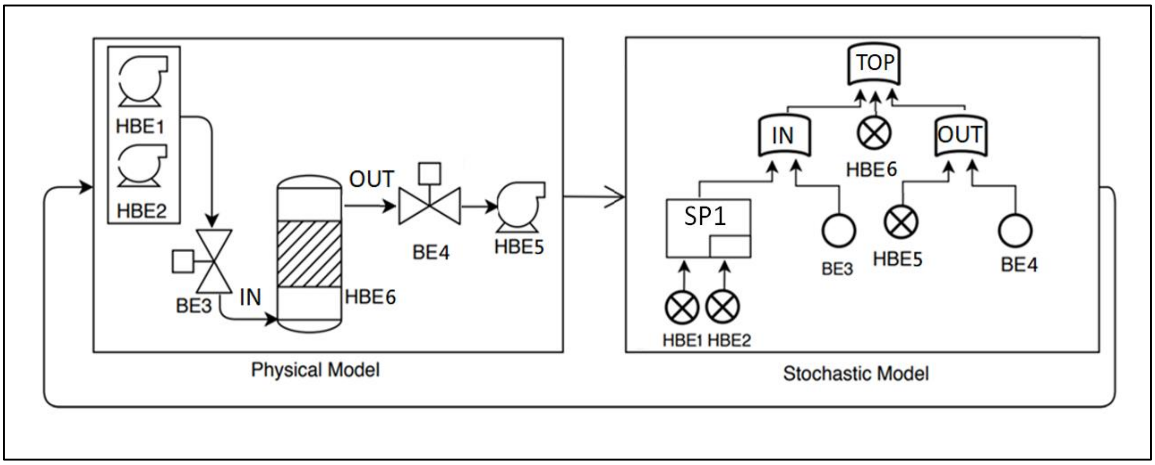


Figure 2. Representation of the Stochastic Hybrid Fault Tree Automaton (SHyFTA) model of the distillation column case study.

This system is constituted by a distillation column, working continuously 24 h per 365 days a year, that has to purify the output mixture of an industrial process, separating the solid part from the liquid. The undesired top event corresponds with the unavailability of the system, which involves the stopping of the purifying process. The input section of the distillation column (identified with the subsystem IN) is constituted by two electric pumps (HBE1 and HBE2) and by a valve (BE3) in charge of pumping the mixture to purify to the distillation column. The electric pumps are in a standby configuration; therefore, they are modelled with a SPARE gate (SP1). The suction system (OUT) of the distillation column is made up of a valve (BE4) and by an electric pump (HBE5). Moreover, it is assumed that the tank containing the solid part needs to be cleaned when it is full. This event is represented by the hybrid basic event HBE6. As shown in the fault tree, the top event is verified if any among the IN, OUT or HBE6 occurs. For this reason, an OR gate (TOP) is used.

The electric pumps (HBE1, HBE2, and HBE5), modelled as hybrid basic events, are characterized by a dynamic failure rate according to the formula of the MIL-HBDK-217 standard [58]. This standard has the great merit to identify a mathematical relationship between the failure rate and the aging L. Therefore, even the correctness of this relationship cannot be guaranteed by the authors. It represents a valuable example for illustrating how to use the SHyFTA methodology.

$$\lambda(t) = (\frac{L^2}{\alpha_B^3} + \frac{1}{\alpha_W}) \tag{4}$$

where α_B is the characteristic life of the bearings and α_W is the characteristic life of the coils. Instead, the hybrid event HBE6 depends on the physical characteristics of the solid tank of the distillation column and on the physical characteristics of the input mixture to separate. In fact, the volume of the solid part can be computed by the following formula:

$$Vs = \int_{0}^{t} S(\tau)\cdot Q(\tau)d\tau \tag{5}$$

where $S(\tau)$ is the fraction of solid part contained in the flow mixture $Q(\tau)$, input of the distillation column. When the volume of the solid part deposited in the tank reaches the Vs threshold of the tank (V_{ST}), $Vs = V_{ST}$, the distillation process must be stopped to perform the operation of cleaning of the solid tank of the distillation column.

Moreover, a quality requirement of the cleaning process specifies that the liquid part achieves the standard purity if, after a stop of the system, the process restarts within 16 hours. In other words, the mixture inside the distillation column cannot lie idle longer than 16 hours, otherwise it must be dumped. Finally, at the end of the year, the company has to determine the amount of mixture that could not be purified. If it exceeds a certain threshold, the safety of the environment is compromised, and an economic penalty has to be paid.

The process described so far contains many elements that cannot be modelled with any of the modelling “Traditional Static” or the “Advanced & General-Purpose” methods. In fact, the dynamic failure rate specified in Equation (4), the condition for triggering the HBE6 in Equation (5) and the constraints concerning the quality requirement and the safety of the environment have to be modelled considering the dynamics of the physical process. A SHyFTA model of the presented case study is developed using the SHyFTOO, and the next subsections contains the main information for implementing it. Several information that will be presented refer to the main components of the SHyFTOO library that are discussed, thoroughly, in the Appendix A. To improve the reading of the manuscript, throughout the manuscript, the elements of the SHyFTOO library that are part of the Appendix A are referred with the following notation: << *component* >>.

3.1. Definition of the Fault Tree

The definition of a DFT requires the customization of the <<*initFaultTree.m*>> script. As is shown in Algorithm 1, the code for the specification of the fault tree model is intuitive.

Algorithm 1. Code for the definition of a fault tree using the SHyFTOO library

```
%% Define the Fault Tree Structure %%
Tm = 8760; %[h]
%% Define BEs %%
%input pumping system
HBE1 = BasicEvent(’HBE1’,’hybrid’,’exp’,[],[1/135]); %pump 1
HBE2 = BasicEvent(’HBE2’,’hybrid’,’exp’,[],[1/135],’exp’,[5e-07]); %pump 2
BE3 = BasicEvent(’BE3’,’exp’,’exp’,[1.19e-4],[1/135]); %valve 1
%output suction system
HBE5 = BasicEvent(’HBE5’,’hybrid’,’exp’,[1.64e-4],[1/135]); %pump 3
BE4 = BasicEvent(’BE4’,’exp’,’exp’,[1.19e-4],[1/135]); %valve 2
% sedimentation Basic Event
HBE6 = BasicEvent(’HBE6’,’hybrid’,’exp’,[],[1/3]); %sedimentation hbe
% %% Define Gates %%
SPARE1 = Gate (’SPARE1’, ’SPARE’, false, [HBE1], [HBE2]);
OR1 = Gate (’OR1’, ’OR’, false, [BE3, SPARE1]);
OR2 = Gate (’OR2’, ’OR’, false, [HBE5, BE4]);
SHATOP = Gate (’TOPOR’, ’OR’, false, [OR1, OR2, HBE6]);
TOP = SHATOP;
```

Line 2 defines the mission time Tm, whereas from line 5 to 18 the fault tree components are set. Initially, the basic events to be used as parameters of the gate definition are defined. Line 18 is very important because the SHyFTOO library expects to recognize a Top Event by the use of a variable TOP that is equals to the top event gate defined by the user.

3.2. Definition of the Physical Process (with Simulink)

As already anticipated, the Simulink toolbox simplifies the design of complex dynamic systems. Thanks to the SHyFTOO library, it can be easily coupled with the stochastic process by accessing the parameters of the DFT components that must be incorporated within the physical process (e.g., the status of a BE). The Simulink implementation of the physical process for the case study proposed is shown in Figure 3.

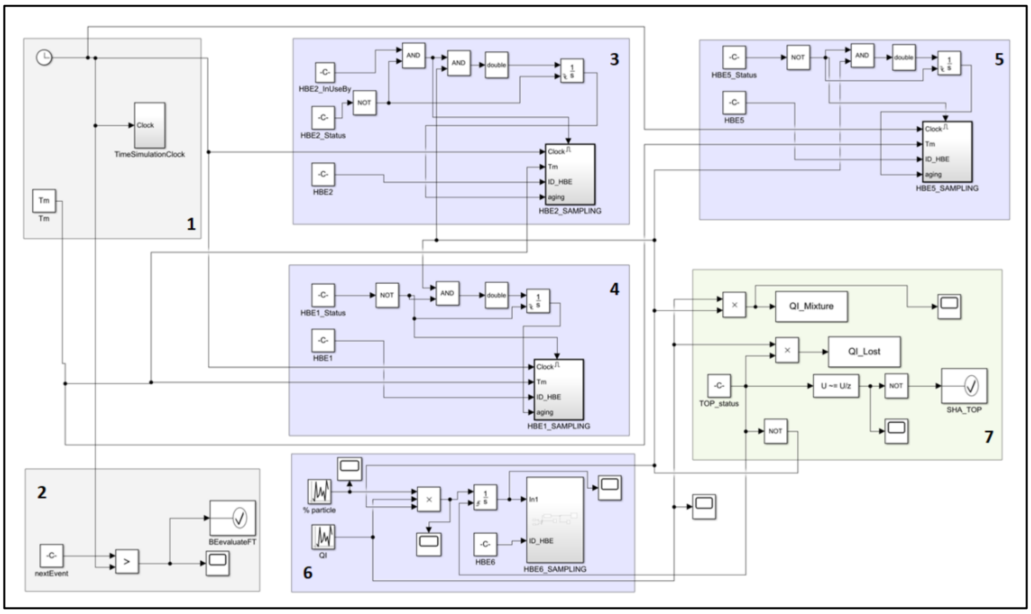


Figure 3. Simulink model of the physical process of the case study.

Among the displayed blocks, blocks 1 and block 2 belong to the standard built-in blocks of the SHyFTOO Simulink (presented in the Appendix A). They are respectively the <<*ITER EVOLUTION* >> and <<*RACE CONDITION* >> and they must be included without any customization in the Simulink implementation of the physical process. Blocks 3, 4, and 5 inherit from the <<*GENERIC HYBRID BASIC EVENT*>> block, and only a few modifications are needed to adapt the internal <<*HBE_SAMPLING*>> block to the characteristics of the probability density function of the corresponding events. Block 6 models the physical process that determines the triggering of the HBE6. In fact, it depends on the condition $Vs = V_{ST}$, knowing that Vs follows Equation (5). Finally, block 7 models the quality requirement of the cleaning process, which is not a dependability attribute but a physical property that depends on the availability of the distillation column.

To better understand the custom modification of all these blocks, in the next subsections, the Simulink models are discussed into more detail.

3.2.1. Customization of the HBE1 (and HBE5) Block

The Simulink ensemble shown in Figure 4 is devoted to the failure sampling of the hybrid basic events HBE1, identified with the label ‘4’, and with the same logic to the HBE5, identified with the label ‘5’.

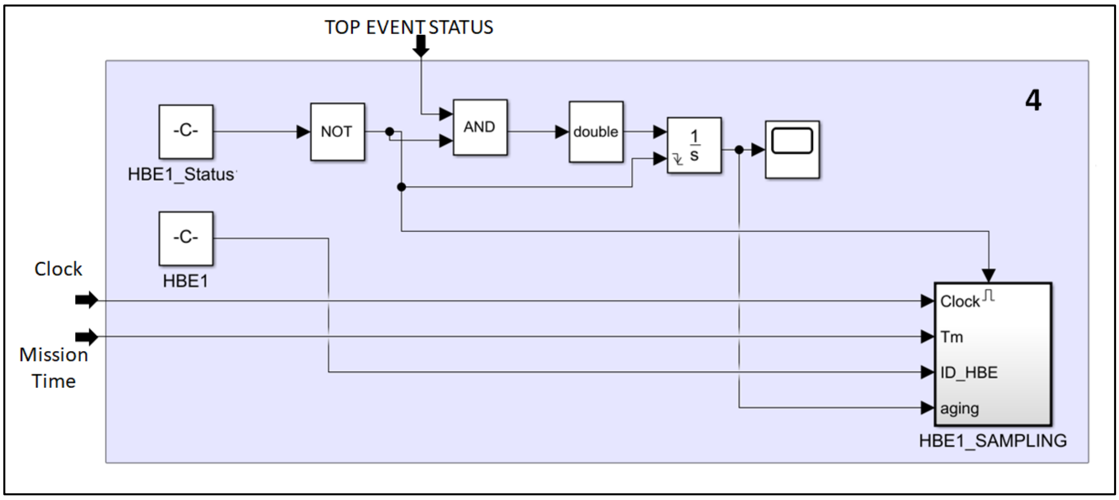


Figure 4. Simulink model of the HBE1 and HBE5 of the case study.

The block HBE1_SAMPLING contains the logic to simulate, at any timestep, the failure of the electric pump according to the dynamic failure rate specified in Equation (4). This equation depends on the aging of the electric pump, and one of the inputs to provide to the "*Interpreted Matlab function*" is the aging variable. As shown in Figure 4, the *aging* is computed using an integrator block that takes as input a Boolean signal (converted into a double value), the output of a complex logic that considers the status of the electric pump itself (<<*HBE1.Status*>>), and the status of the Top Event. If both are good, it means that the pump is operating correctly, so its *aging* grows linearly. Clearly, if the pump has failed, the *aging* can no longer increase. With the same reasoning, the aging does not increase even if the Top Event has occurred because it is assumed that, if the cleaning process has paused all, the components stop operating until the system is restored. The integrator block takes another input as a reset condition so as to reset the value of the *aging* to 0 when the pump is restored as new.

3.2.2. Customization of the HBE2 Block

The Simulink ensemble shown in Figure 5 models the failure sampling of the hybrid basic events HBE2 identified with the label '3'.

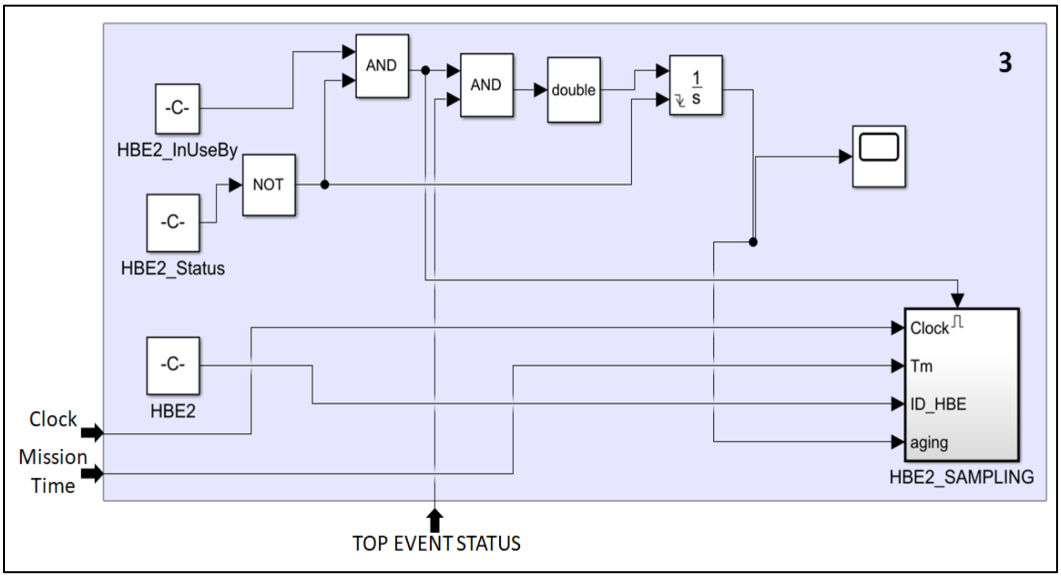


Figure 5. Simulink model of the HBE2 of the case study.

This component is the electric pump in standby position with the HBE1 under the spare gate SPARE1 of the DFT. When the HBE1 fails, the stochastic model modifies the property '*InUseBy*' of the spare component HBE2 ((<<*HBE2.Status*>>) and sets the value of the identifier index of the served spare gate. This property is used, together with the status of the component and of the Top Event, to control the Simulink logic of the Boolean signal input of the aging integrator. In this way, the aging is increased only if the status of the electric pump itself and the status of the Top Event (<<*Top.Status* >>) are good and, if and only if, the property <<*HBE2.InUseBy*>> takes a positive value (corresponding with the index of the spare gate served).

3.2.3. Customization of the HBE6 Block

The Simulink ensemble shown in Figure 6 is devoted to the failure sampling of the hybrid basic events HBE6, identified with the label '6'.

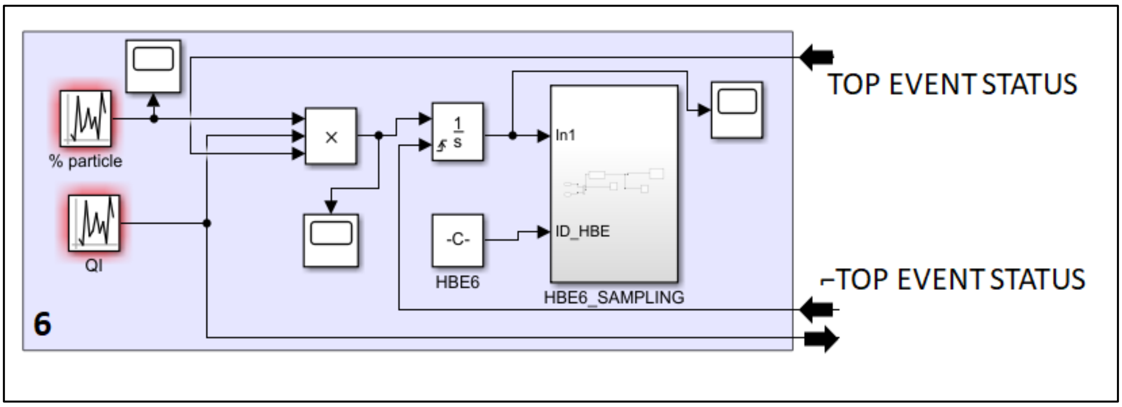


Figure 6. Simulink model of the HBE6 of the case study.

The blocks "% particle" and "QI" represent respectively the two physical variables constituting the input mixture to the process, where it is assumed that the former is the percentage of the solid part inside the infinitesimal volume *QI*, constituting the instantaneous flowrate of mass processed in the distillation column. Those are random variables that vary at any timestep of the simulation with a uniform probability density function distribution. It was assumed that the percentage of solid part can take at most 2% of the instantaneous *QI* mass. The *QI* vary in the range [400, 600] m^3/h.

Back to Figure 6, it is possible to see that these two variables are multiplied and integrated so as to implement the formula of Equation (2). The product block takes as input also the <<*TOP.Status*>> because, if the system is unavailable, it is assumed to stop the cleaning process with a loss in terms of mixture mass, which never stops flowing (in that case it is lost and must be recorded to evaluate the environmental damage). Finally, the *HBE6_SAMPLING* block, shown in Figure 7, modifies a bit the template model of Figure A2 (see Appendix A) as the event modelling the filling of the solid tank is not governed by a stochastic probability density function, but by a physical condition. Therefore, the "*Interpreted Matlab function*" to use in this block has been codified ad-hoc, as shown in Algorithm 2.

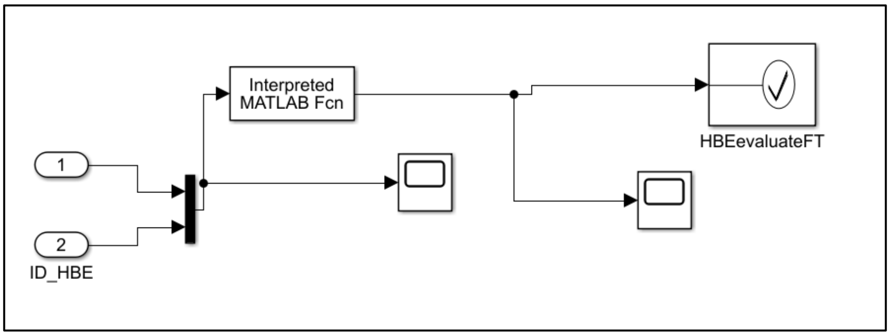


Figure 7. The Simulink implementation of the hybrid basic event HBE6.

Algorithm 2. Code for the definition of the failure logic of the HBE6.

```
1:   function y = HBE6_failure(u)
2:       coder.extrinsic('evalin', 'assignin')
3:       % Read value from workspace
4:       currentTime = get_param(bdroot,'SimulationTime');
5:       FTA = evalin('base','FT');
6:       HBEid = getID(FTA,strcat(u(2:end-1))'); %get the id of the Basic Event
7:       decanterCapacity = evalin('base','decanterCapacity');
8:       y = 1;
9:       if(u>=decanterCapacity)
10:          assignin('base','indexTriggeringBE', HBEid);
11:          FTA(HBEid).FailureTime = currentTime;
12:        assignin('base','FTA',FTA);
13:        y = 0;
14:      end
15:  end
```

Line 2 is a Matlab® API that enables the assignment and the evaluation of the workspace variables (e.g., the global variables of the main script <<*SHyFTAmain.m*>>) within a Matlab® function. For instance, in line 5, the variable *FTA* is initialized with the value of the workspace variable *FTA* that contains all the element of the stochastic model, including basic events and gates. In line 6, the variable *HBEid* is initialized with the identifier of the hybrid basic events whose name is contained in the input u (2), passed to the function. Line 7 initializes the local variable *decanterCapacity* with the value set in the script <<*SHyFTAmain.m*>> and it represents the capacity V_{ST} of the solid tank. From line 9 to 13, the function performs the comparison of the current volume at time t passed as input u (1) and the value of the solid tank (*decanterCapacity*). If the inequality is verified, the function sets the property <<*FailureTime*>> of the hybrid basic event with the value of the current simulation time, *currentTime*, that was retrieved in line 4 using the Matlab® API get_param (). Afterward, in line 12, the global variable *FTA* is set with the local variable *FTA* that contains the change on the hybrid basic event declared as failed. In line 13 the output of the function is set equal to 0. This value is returned and used in the block assertion <<*HBEevaluateFT*>> that pausing the iteration performs the evaluation of the stochastic model. If the returning value y is equal to 1, the assertion block does not raise, and the iteration can continue without interruptions.

3.2.4. Customization of the Physical Process

The Simulink ensemble shown in Figure 8 models the part of the physical process devoted to the measure of the quality requirement of the cleaned liquid.

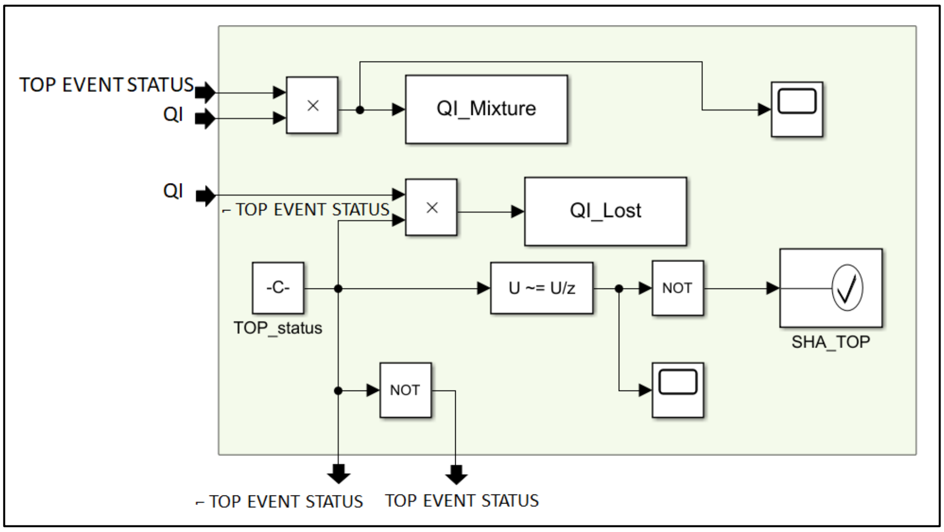


Figure 8. The Simulink implementation of the physical process of the case study.

The output variable *QI_Mixture* records the flow of mixture which is correctly processed. It is obtained multiplying the input *QI* with the <<*Top.Status*>>. In other words, the variable *QI_Mixture* will record the value 0 for those instants of time in which the system is not working (i.e., namely it is unavailable). On the other hand, with the same logic, the output variable *QI_Lost* records the flow of mixture which cannot be processed during the system unavailability. This value is obtained multiplying the input variable *QI* with the complement to one of the <<*Top.Status*>> *(= "¬TOP EVENT STATUS")*.

The other part of the ensemble implements the logic to verify the quality requirement of the liquid part, separated in the distillation column. It makes use of the <<*Top.Status*>>. Any change from bad to good and vice versa triggers an assertion (*SHA_TOP*) that verifies if the amount of time from the last stop (due to the unavailability of the system) exceeds the threshold dictated by the quality process variable *time4GoodProcess*. The following code, shown in Algorithm 3, is used. At first, it can be noticed that, in Matlab®, the script has complete visibility of the workspace global variables and therefore it is not required any coder.extrinsic API to access them. Line 2 checks if the Top Event has occurred and assigns to the global variable *lastStop* the value of the current time, which has been retrieved using the *get_param* API of Matlab®.

Algorithm 3. Code for the definition of the quality requirement of the case study.

```
1:   currentTime = get_param(bdroot,'SimulationTime');
2:   if(TOP.Status == 1)
3:       lastStop = currentTime;
4:   else
5:       if(currentTime - lastStop > time4GoodProcess)
6:           if(lastStop>0)
7:               volumeNok = volumeNok+QI_Mixture.Data(lastStop);
8:           else
9:               volumeNok = volumeNok+QI_Mixture.Data(lastButOneStatusChange);
10:          end
11:      end
12:  end
```

Otherwise, when the system has recovered, the algorithm checks the time needed to restart the process, obtained as the difference between the current time and the last stop variable. If this value is longer than the constant process variable *time4GoodProcess* (that specifies the max amount of time that the mixture can be idle to remain good) then the global variable *volumeNok*—keeping track of the volume of mixture so far dumped—has to be updated.

In order to record the other relevant physical, the script <<*shyftaMetrics.m*>> is used. Since, the <<*shyftaMetrics.m*>> is launched at the end of each iteration, it is possible to compute the volume of mixture cleaned and the volume of mixture that could not be processed because of the unavailability of the system using the Matlab® function *sum*, passing as parameter the arrays of the *QI_Mixture* and *QI_Lost* set in the Simulink block shown in Figure 8.

4. Case Studies Resolution

This section contains the results of several case studies, including the one described in Section 3, selected to demonstrate the applicability of the SHyFTA methodology to a wide range of industrial applications. The resolution of these case studies has been carried out with the SHyFTOO library, but the corresponding models are not included for space limits.

4.1. Electric Motor

Figure 9 shows the simplified SHyFTA model of an electric motor. This system consists of several components where, among all, it is possible to identify the BE1 and BE3 (bearings), and BE2 (winding or coil), characterized by the dynamic failure rates of Equations (6) and (7).

$$\lambda_{BE1} = \lambda_{BE2} = L^2(t) \cdot \left[10^{\left(2.534 - \frac{2357}{T_A + 273}\right)} + \frac{1}{300 + 10^{\left(20 - \frac{4500}{T_A + 273}\right)}} \right]^3 \tag{6}$$

$$\lambda_{BE3} = \frac{1}{10^{\left(\frac{2357}{T_A + 273} - 1.83\right)}} \tag{7}$$

In Equations (6) and (7), $L(t)$ represents the aging of the component, and T_A is the ambient temperature. The parameters of these equations are taken from the standard MIL-HDBK-217 [58]. This standard has the great merit to identify a mathematical relationship between the failure rates and the real-time variables of the aging and of the ambient temperature; even the correctness of this relationship cannot be guaranteed by the authors, it represents a valuable example for illustrating how to use the SHyFTA methodology. Unfortunately, we could not find a different data source that could have allowed us to perform a double check of the final results.

The failure of one of these components causes the failure of the entire system. To account for further types of failure, a generic BE4 event has been included in the fault tree model. As it is possible to see in the Figure 9, BE4 is the only basic event modelled as a traditional basic event.

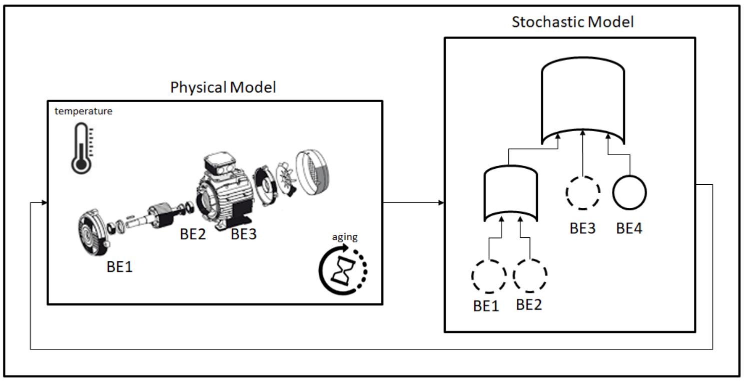


Figure 9. The SHyFTA model of an electric motor.

The main goal is to assess the reliability of the electric motor and compare it with the one obtained using a traditional FTA model. Due to its limitations, the static FTA cannot take as input a dynamic failure rate and therefore the parameters shown in Table 2 have been fixed with constant values using the method of computation described in the [58]. Moreover, for λ_{BE1} and λ_{BE2}, the worst-case failure rates (with $L = T_m = 8760$ h) have been used.

Table 2. Failure and repair rates of the Fault Tree model of the electric motor case study.

Event	Failure Rate (MIL-HDBK-217)
BE1/BE2	5.98×10^{-6} (h^{-1})
BE3	5.89×10^{-7} (h^{-1})
BE4	1.00×10^{-6} (h^{-1})

Figure 10 shows the unreliability results for the two models—the Fault Tree and the SHyFTA, respectively. It is possible to notice that the unreliability computed with the Fault Tree is higher than the one simulated with the SHyFTA model. The former increases linearly whereas the latter presents a pronounced non-linear behaviour after the 7000th hour from the beginning of the mission.

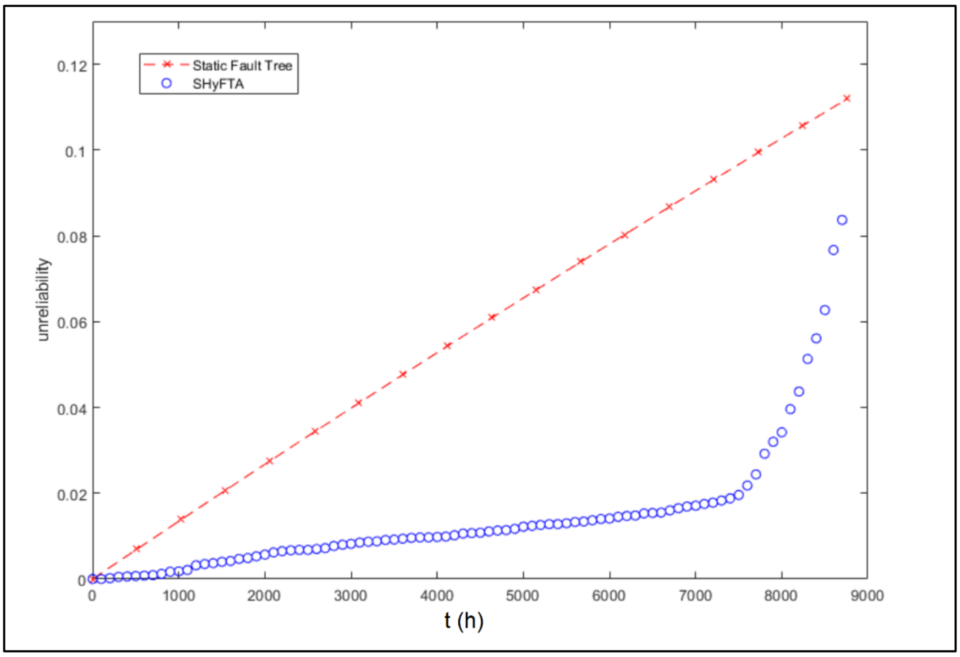


Figure 10. Unreliability of the electric motor and comparison with the static Fault Tree model result.

To better understand the effect of the bearing over the unreliability of motor, it is important to recall that its failure rate increases with the square of the aging L and decreases with the cube of α_B as shown in Equation (6). On the other hand, α_B grows with the increasing of the ambient temperature whose trend is shown in Figure 11. The combination of these effects on the dynamic failure rate of the bearing can be analysed by observing the trend in Figure 12. At the beginning of the mission, the failure rate is low, although around the 1000th hour, it presents an interesting peak of about 3.5×10^{-5} caused by a sudden decrease of the ambient temperature in wintertime.

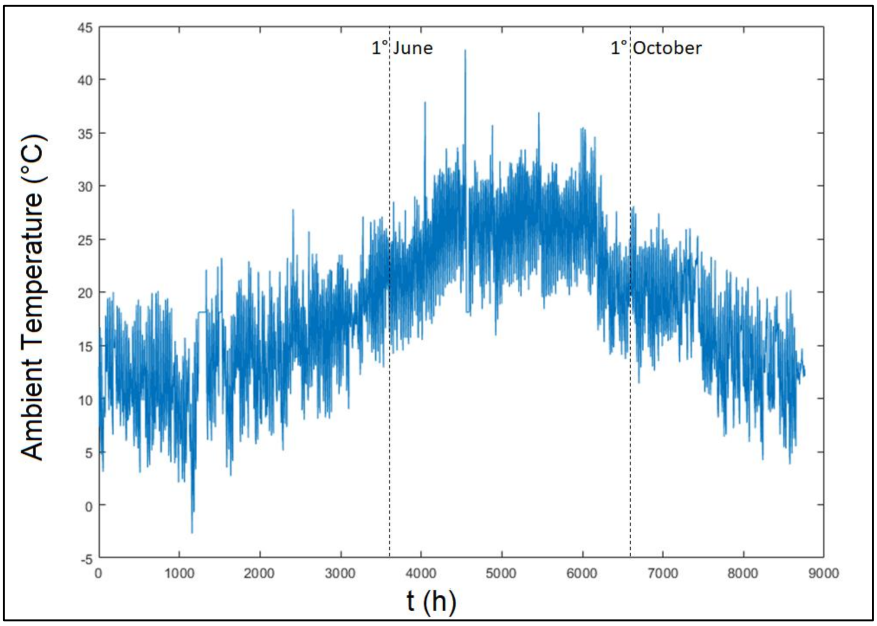


Figure 11. Ambient temperature used as input of the SHyFTA model.

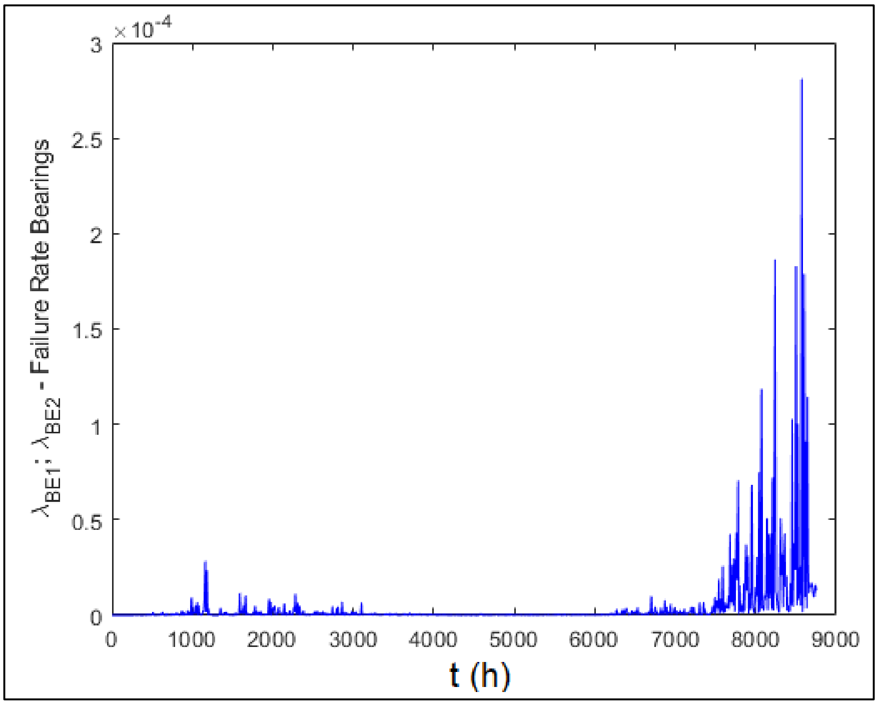


Figure 12. Failure rate of the bearings of the electric motor case study.

During the hot season (indicatively from the 1 June–1 October, in the interval comprised between the 3624th and the 6552th h) the failure rate is constant because of the prevalence of the parameter α_B over the aging L. However, when the ambient temperature starts decreasing again (after 1 October), the aging effect begins to dominate and the failure rate presents several peaks that are revealed during the night-time hours, when the ambient temperature decreases. This explains the important spike of the unreliability of the motor, after the 7000th h. These results demonstrate the huge difference between a static model and a hybrid dynamic reliability model, with this latter able to model and capture non-trivial physics-based dynamic operation features of a real electro-mechanical process.

4.2. Domestic Photovoltaic Power Plant with Storage System

In this case study [59], a SHyFTA model of a domestic grid-connected photovoltaic power plant has been simulated in order to compare the performance and the economic benefits of the same configuration, with and without a storage system. The main technical information of the system is shown in Table 3. The single-line diagram of the grid-connected photovoltaic power plant is shown in Figure 13.

Table 3. Main characteristics of the photovoltaic power plant.

Description	Value
PV Nominal Power	3000 W
PV Panel Nominal Power	300 W
Panel efficiency (η)	15%
Inverter Nominal Power	2.8 kW
Inverter Efficiency (η)	97%
MPPT range	210–550 V
Battery Capacity	6.4 kWh
Maximum charging/discharging Current	64 A

The SHyFTA model has to simulate the process of energy supply for a generic household equipped with a grid-connected photovoltaic power plant and a storage system. The single-line diagram, in Figure 13, allows the identification of the main sub-systems: the photovoltaic power plant (PV Generator), the storage system (BAT), and the equipment of the grid connection coupling (GCC) that allow the coupling with the electrical grid.

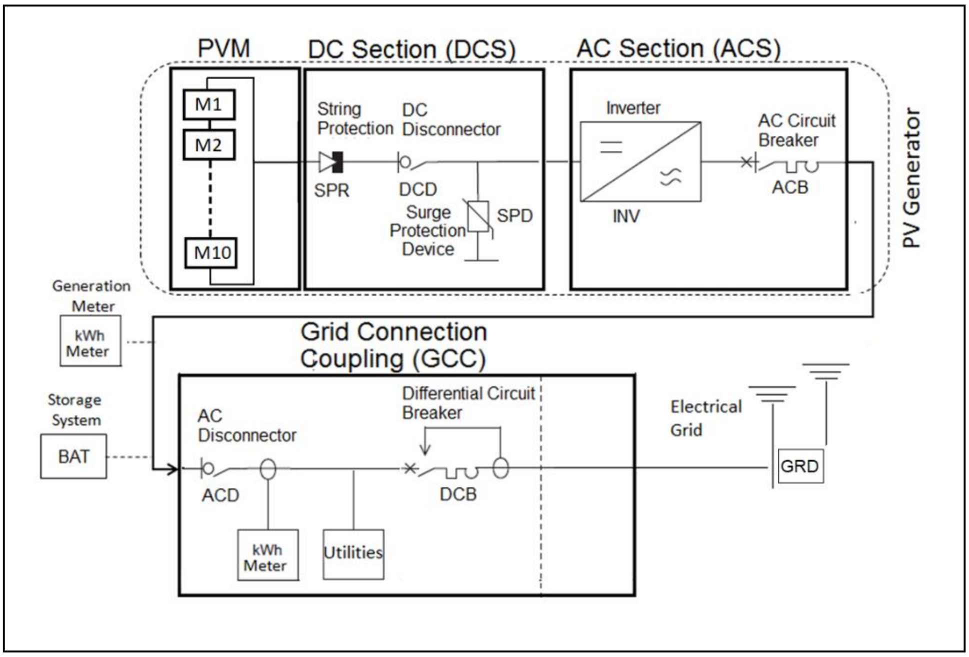


Figure 13. Schema of the deterministic process of the PV system, including the battery.

It is possible to identify the following main sub-sections:

1. PV Module (PVM), made up by ten photovoltaic modules (M1–M10);
2. Direct Current Section (DCS), made up of string protection diodes (SPR), a DC disconnector (DCD), and a surge protection device (SPD);
3. Alternating Current Section (ACS), made up of an inverter (INV) and an AC circuit breaker (ACB);

4. Grid Connector Coupling (GCC), made up of an AC disconnector (ACD), a differential circuit breaker (DCB), and a generic sub-system representing the electrical grid (GRD).
5. Battery (BAT) that is connected in parallel in the AC section.

The technical regulation of the grid-connected photovoltaic [60] power plant states that if the electrical grid fails, the power plant must be disconnected, stopping the production and the energy supply of the household. In this case, and any time the photovoltaic power plant is disconnected, the battery is also forbidden to supply energy to the household. This scenario corresponds with the top event of the DFT of Figure 14. The DFT is constituted by the Top Event AND gate (TE) that takes as input the output of the OR gate GCC (OR (ACD, DCB, GRD)) and the OR gate "PV Down" (OR (ACS, DCS)). The former models any type of disconnection of the electrical grid, whereas the latter the unavailability of the photovoltaic power plant that occurs if the electrical circuit of the PV Generator gets open (any failure of the ACS or DCS components). The AND Gate PVM models the failure of the photovoltaic strings; although the modules are connected in series, the by-pass diodes guarantee the electrical isolation of those modules that are not working properly. As far as it concerns the battery, it must be pointed out that its unavailability does not cause a stop of the household energy supply because the grid can fulfill the energy request. On the other hand, the battery is disconnected if the PV power plant is down. These behaviors can be modelled with two FDEP Gates. The first takes as primary input the output of the GCC and as secondary input the OR Gate "PV down" [59]. The second FDEP takes as primary input the "PV Down" output and as second input the gate "BAT down". The failure and repair rates of the DFT model are shown in Table 4.

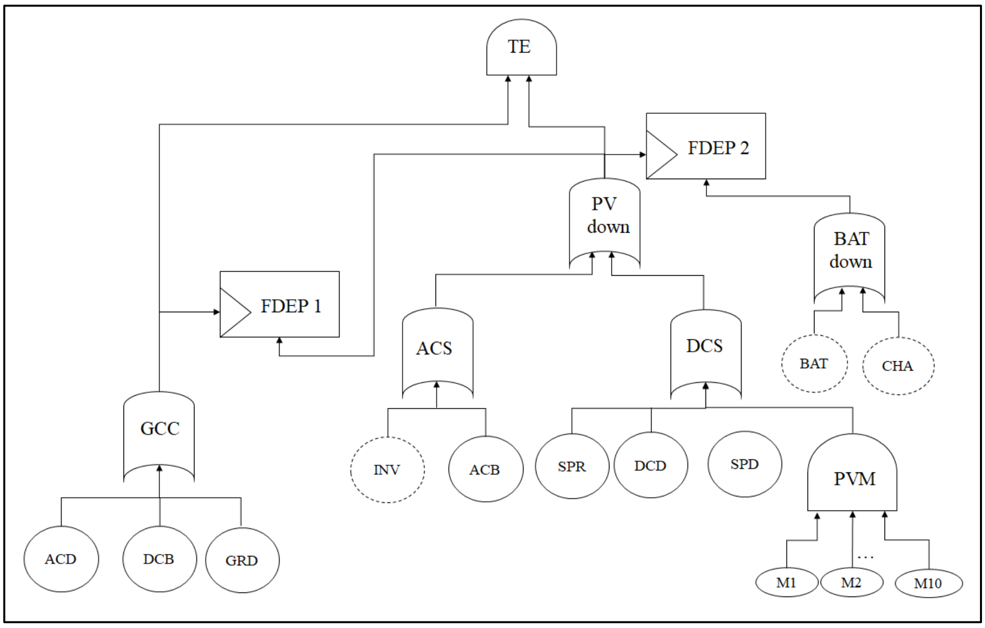


Figure 14. Dynamic fault tree of the domestic photovoltaic power plant equipped with a storage system.

The inverter and the battery are the most sensitive components to the aging. Therefore, they have been modelled with a dynamic failure rate that allows to consider the wearing-out. To this end, a Weibull pdf with shape factor $\beta > 1$ (i.e., the failure rate is increasing with respect to time) [54] and a scale parameter γ (that is generally set with the corresponding constant failure rate of an exponential distribution) has been used. Equation (8) defines the failure rate as

$$\lambda(t) = \beta/\gamma \cdot (L(t)/\gamma)^{\beta-1} \quad (8)$$

The aging variation is modelled using a piecewise Markov process, with the following equation:

$$\frac{\mathrm{dL}}{\mathrm{dt}} = \mathrm{i}_{\mathrm{on}} = \begin{cases} 1, & \text{if the component is switched on} \\ 0, & \text{if the component is switched off} \end{cases} \tag{9}$$

Table 4. Failure/repair rates and of the photovoltaic power plant.

	Component	Failure/Repair Distribution	λ: Failure Rate (y^{-1})	μ: Repair Rate (h^{-1})
Mi	PV Module	Exp/Exp	1/15	1.4×10^{-3}
SPR	String Protection	Exp/Exp	1/10	1/48
DCD	DC Disconnector	Exp/Exp	1/3	1/12
SPD	Surge Protection	Exp/Exp	1/10	1/48
INV	Inverter	Wei/Exp	See Equation (5), $\beta = 1.5, \gamma = 40.29 \times 10^{6}$	2.1×10^{-3}
ACB	AC Circuit Breaker	Exp/Exp	1/10	1/12
ACD	AC Disconnector	Exp/Exp	1/3	1/12
DCB	Diff. Circuit Breaker	Exp/Exp	1/3	1/48
GRD	Grid	Exp/Exp	5	1/4
BAT	Storage System	Wei/Exp	See Equation (5), $\beta = 1.5, \gamma = 1.14 \times 10^{5}$	2.1×10^{-3}
CHA	BAT Energy Shortage	This event is modelled in the physical process. See [59] for more info.		

The economic assessment has been carried out using the cash-flow method along the 20 years of operations, considering the main characteristics of the Italian Market. In order to compute the net present value (NPV) of the investments, it was required that we evaluate the service availability. This attribute of the dependability assessment depends on the failure/repair behaviour described by the DFT of Figure 14 and also on the capability of the system to provide the expected service when it is needed. This latter corresponds with the energy provisioning. Therefore, in this case study, the hourly household energy demand (Table 5) must be provided as input to the SHyFTA model.

Table 5. Seasonal daily consumption of a household case study.

H	Autumn (Wh/h)	Spring (Wh/h)	Summer (Wh/h)	Winter (Wh/h)
00:00	289.3	39.3	289.3	539.3
01:00	39.3	39.3	39.3	39.3
02:00	289.3	39.3	39.3	39.3
03:00	39.3	39.3	289.3	39.3
04:00	289.3	39.3	689.3	439.3
05:00	39.3	439.3	289.3	39.3
06:00	774.3	39.3	574.3	824.3
07:00	599.3	324.3	349.3	599.3
08:00	39.3	599.3	539.3	39.3
09:00	39.3	539.3	439.3	439.3
10:00	39.3	439.3	689.3	439.3
11:00	39.3	439.3	39.3	39.3
12:00	39.3	39.3	289.3	39.3
13:00	39.3	39.3	39.3	39.3
14:00	39.3	39.3	539.3	39.3
15:00	39.3	289.3	539.3	39.3
16:00	439.3	289.3	939.3	439.3
17:00	764.3	689.3	764.3	264.3
18:00	824.3	514.3	1164.3	764.3
19:00	2224.3	914.3	1824.3	1824.3
20:00	1574.3	1574.3	1574.3	1824.3
21:00	574.3	1324.3	574.3	824.3
22:00	349.3	324.3	349.3	599.3
23:00	39.3	99.3	289.3	539.3
Daily Avg(Wh/y)	9463.2	9153.2	13153.2	10753.2

Comparing the two design configurations, with and without storage system, the results of the SHyFTA model allowed to estimate the (1) expected energy production that is injected to the grid (Figure 15) and the (2) energy required (from the grid) to satisfy the energy consumptions of the household (Figure 16).

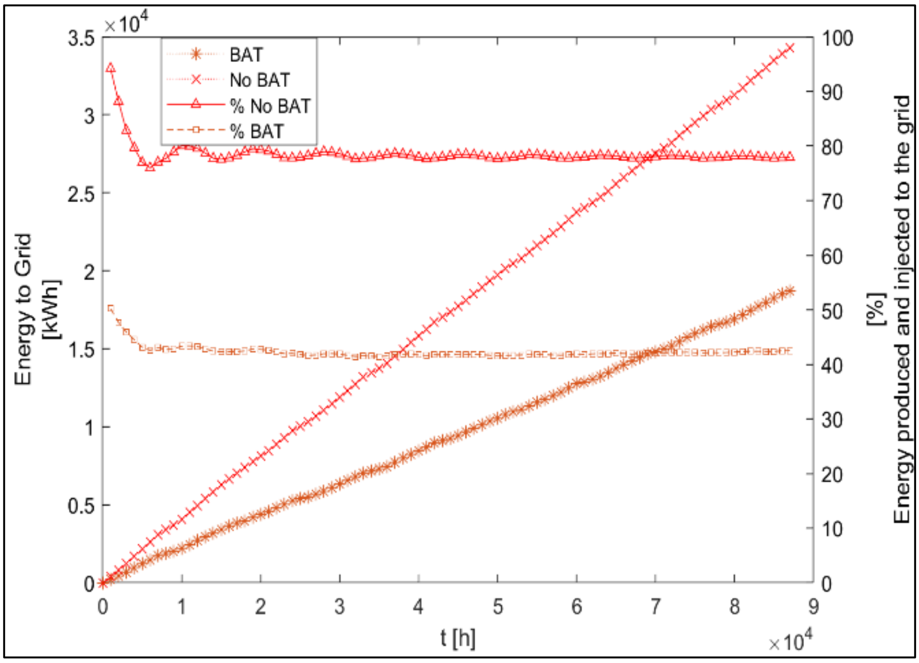


Figure 15. Expected progressive energy production injected to the grid.

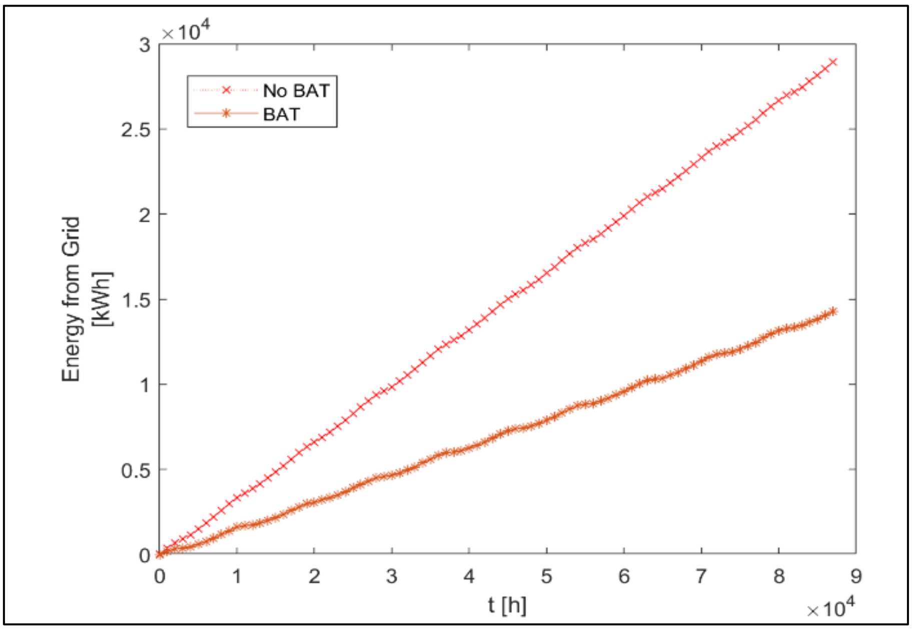


Figure 16. Expected progressive energy required from the grid.

These two variables depend on the profile of the hourly energy demand of Table 5. Clearly, the investment of the PV system (in both the configurations) reduces the energy demand required from the grid of the household and has an impact into the electrical bill. This latter information, together with the cost of the investments, is used to perform a cash-flow analysis that, in this example, has been carried out considering the Italian market scenario.

Figures 17 and 18 show the results of the economic analysis. In particular, Figure 17 presents the cumulated discounted cash-flow along the investment horizon time and Figure 18 the payback time (in years) as a function of the battery cost (expressed in €/kWh).

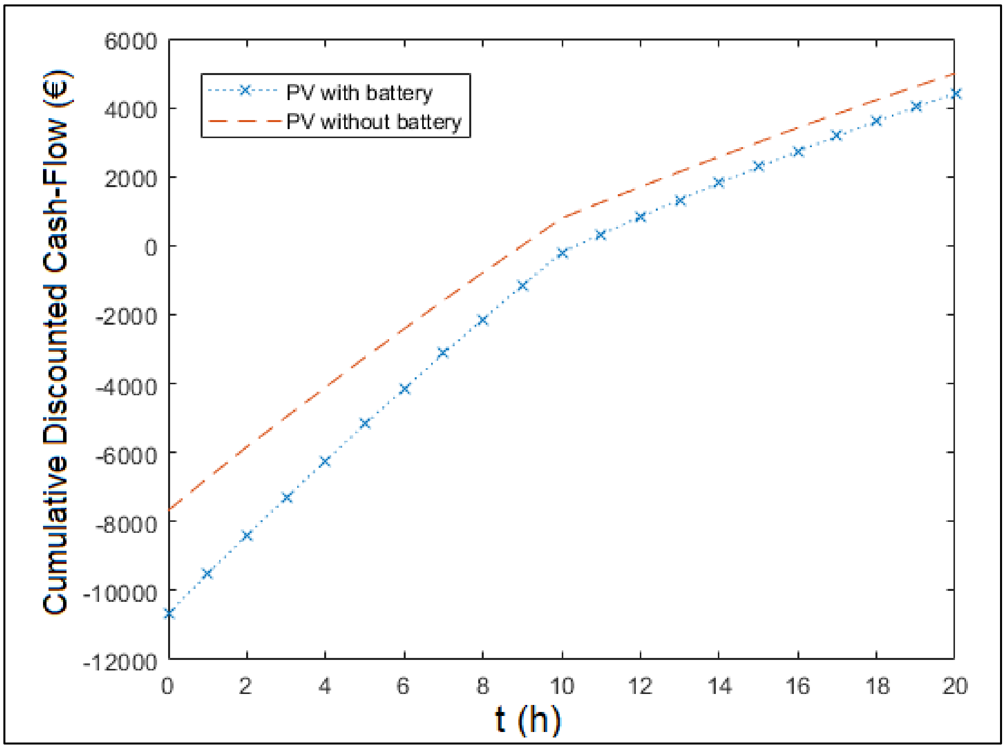


Figure 17. The return of investment.

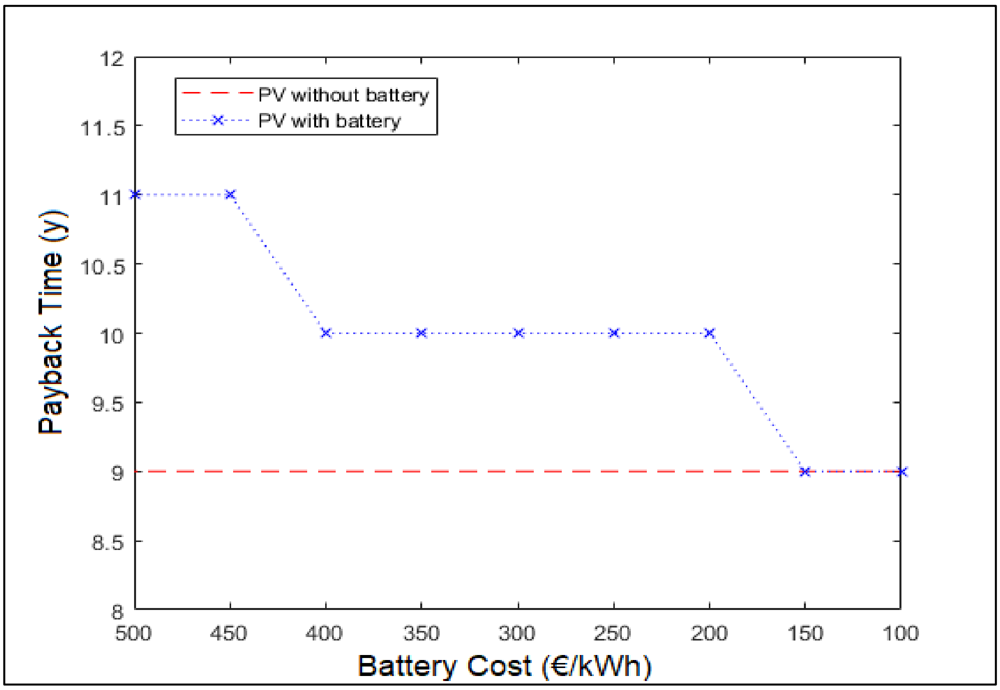


Figure 18. Payback time sensitivity analysis depending on the variation of the battery cost.

In Figure 17, it is clearly demonstrated that the investment of a PV system without battery is more convenient than an investment of a PV system with battery. In fact, the final NPV (20y, 3%)

of the investment without battery amounts to 4496 €, versus 4410 € of the investment with battery. On the other hand, in Figure 18 it is shown that the payback time of the investment without battery is nine years, whereas with the battery amounts to 11 years. Only a drastic cost decrease of the battery system can modify the results in favor of the battery installation. In fact, as shown in Figure 18, the payback time of the PV system with battery becomes equal only if the cost of the battery is lowered to 150 €/kWh.

4.3. Distillation Column

The results presented in this section refer to the case study discussed in Section 3. Whereas in the previous case studies the SHyFTA was used to evaluate the system reliability, the service availability and the NPV, in this final example the SHyFTA model is coded to dimension the capacity of the decanter that maximizes the yearly profit (Incomes–Costs) along the useful life of the system (T_{UL} = 10 years). First of all, knowing that the cost of installation for a solid tank with a capacity of 0.1 m^3 amounts to $15,000 USD, it is possible to compute the cost of installation for different sizes of the solid tank. In fact, the well-known "0.6 rule" states that the relationship between the increase in equipment cost (C) and the increase in capacity (V) is given by the following scaling law [61]:

$$\frac{C_1}{C_2} = \left(\frac{V_1}{V_2}\right)^m \tag{10}$$

where m denotes the scale coefficient (= 0.6). Therefore, Table 6 resumes the installation costs and the corresponding yearly expense (amortization) spread during the useful life, assuming a discount rate i_r = 4.5% (i.e., PVA = $\frac{1-(1+i_r)^{-T_{UL}}}{i_r}$ = 7.913).

Table 6. Availability of the distillation column.

Solid Tank Capacity (m^3)	Installation Cost ($)	Yearly Expense ($/y)
0.1	15,000	1859
0.3	28,998	3664
0.6	43,952	5554
0.9	56,058	7084
1.2	66,619	8419
1.5	76,163	9625

Therefore, it is possible to extract from the SHyFTA simulation the yearly volume of the mixture processed in a good way (OK), dumped (NOK), or not processed at all (Lost). The simulation results are shown in Figure 19 (the left axis is assigned to the OK, whereas the right to the NOK and Lost volumes).

Knowing that the purified liquid is sold at the price of $ 0.01 /m^3, it is possible to compute the yearly profit and the yearly missed income. Moreover, among the yearly costs, it is necessary to consider a penalty (of $ 50,000 /year) that must be sustained if the safety environment requirements are not fulfilled. In this case, the safety is the probability of the system to avoid, during a year of activity, that the yearly loss of mixture in the environment exceeds a certain threshold, fixed by the authority. Therefore, the SHyFTA model can be coded so as to compute this probability on the basis of the volume of mixture not processed by the distillation column (Volume Lost) during a year of operations and of a maximum threshold (V_{THmax} = 4.85 × 10^5 m^3). Figure 20 shows the unsafety simulated for a year of activity.

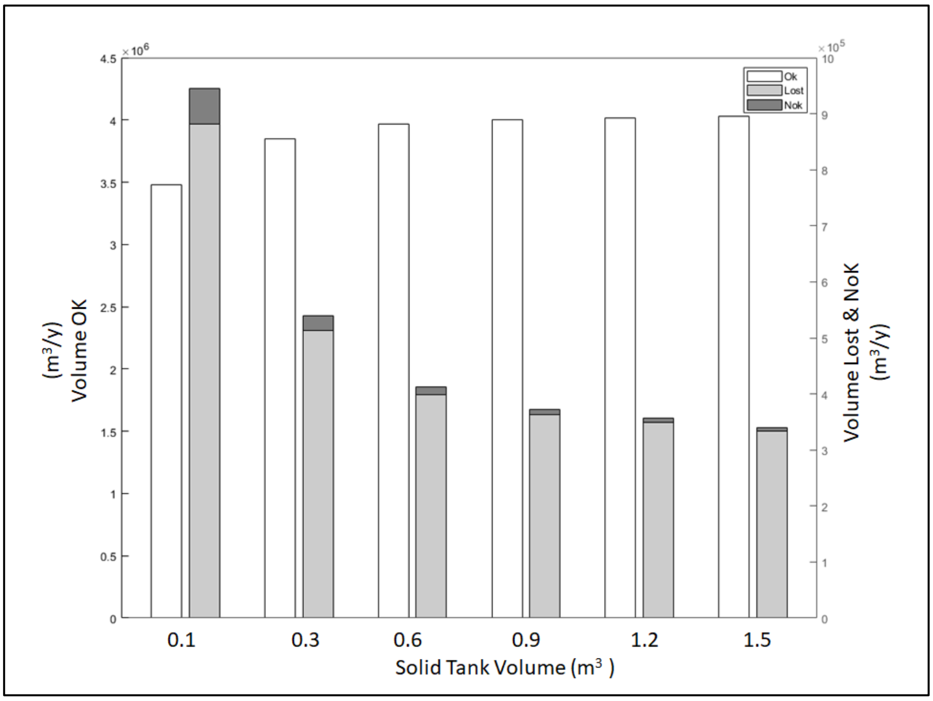


Figure 19. Volume of the mixture processed in a good way (OK), dumped (NOK), or not processed at all (Lost).

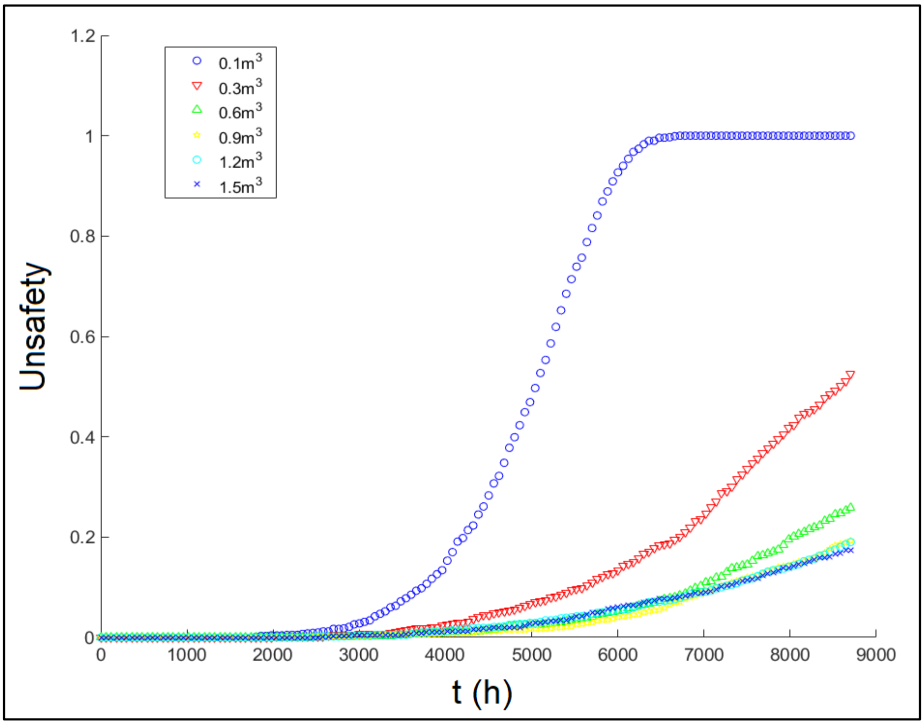


Figure 20. Unsafety of the process of distillation.

It is possible to see that the system with a solid tank of 0.1 m^3 has a probability equals to 1 to exceed the threshold V_{THmax}. Therefore, this solution must be certainly discarded. In order to evaluate the other solutions and account also for the expected penalty, the corresponding unsafety of each solution must be multiplied with the penalty.

Table 7 resumes the values of the yearly profits, expenses and expected penalty for the possible solutions of the solid tank.

Table 7. Profits and costs for several sizes of solid tank.

Solid Tank Capacity (m^3)	Income ($/y)	Missed Income ($/y)	Yearly Expense ($/y)	Safety Penalty ($/y)
0.1	34,815	9457	1859	50,000
0.3	38,497	5390	3664	26,900
0.6	39,643	4127	5554	13,450
0.9	40,006	3720	7084	9900
1.2	40,136	3569	8419	9750
1.5	40,287	3403	9625	8700

Therefore, on the basis of the dependability analysis performed, it is possible to compare the yearly net profit of the various solutions.

As shown in Table 8 the most convenient technical solution, with an annual net profit of $ 19,302/year, is the one with a solid tank of 0.9 m^3 of capacity. According to the results the installation of a larger solid tanks can decrease the unsafety of the system and increase the mean availability but, from the economic viewpoint, is not justified.

Table 8. Results of the dependability assessment.

Solid Tank Capacity (m^3)	Net Profit ($/y)	Mean Availability	Unsafety (at Tm)
0.1	−27131	0.7947	1
0.3	2543	0.8788	0.538
0.6	16,512	0.9050	0.269
0.9	19,302	0.9132	0.198
1.2	18,398	0.9163	0.195
1.5	18,559	0.9197	0.174

5. Conclusions

With the advent of hybrid methodologies, the capability and expressiveness of the stochastic models have increased to such an extent that they require new tools able to assist engineers and risk practitioners in the practice of the dependability assessment. Therefore, the need for a ready-to-use computer-aided tool for the dependability assessment of industrial systems has motivated the research proposed in this paper.

To demonstrate the importance of the topic, a survey of the state-of-the-art has been presented showing the evolution of the Fault Tree Analysis (FTA) related methods. Nowadays, the conception of the Stochastic Hybrid Fault Tree Automaton (SHyFTA) represents a mature alternative to FTA to analyse complex dependable systems under the viewpoint of the Dynamic Reliability Probabilistic Assessment.

The SHyFTOO Matlab® library is a software toolbox that allows for the modelling and the simulation of such hybrid models. The main novelty presented in this paper, over the previous implementation of the SHyFTOO [22], is the capability to couple SHyFTOO with Simulink. Simulink is a toolbox integrated in Matlab®, specifically conceived for the modelling of dynamic systems. This simplifies tremendously the coding of the physical process of a SHyFTA model and provides the user with a toolbox able to compute not only the typical means of the dependability assessment but also other custom performance indexes of the process under analysis.

In this paper, the core concepts of the SHyFTA methodology have been summarized, and the main features and components of the SHyFTOO Simulink library have been presented, in order to offer to the reader a practical step-by-step guide to implement and solve, in an easy way, a SHyFTA model. Afterward, several case studies have been discussed and solved to demonstrate the flexibility of the tool in modelling and solving non-trivial industrial applications. Moreover, the tool allowed to evaluate many properties of an industrial process that cannot be described with non-hybrid methodologies, thus obtaining results and process information that would not be achieved with a different approach.

In future studies, the authors aim to improve the performance of the tool investigating the possibility to integrate it with an algorithm for the variance reduction and for the implementation of rare-event approximations.

The software library developed in Matlab, including some SHyFTA case studies can be freely downloaded from the github webpage of the corresponding author at the following link: https://github.com/chiacchiof/SHyFTOO-Matlab.

Author Contributions: F.C. and D.D. conceived the software tool and the Simulink library; J.I.A. conceived and solved the main case study; S.M.K. solved the first case study, helped with the state of the art survey and with the validation of the Simulink library; L.C. reviewed the methodology, validated the models participated to the review process.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BDMP	Boolean Driven Markov Process
BE	Basic Event
BN	Bayesian Network
CTMC	Continuous Time Markov Chain
DFT	Dynamic Fault Tree
DPRA	Dynamic Probabilistic Risk Assessment
DRBD	Dynamic Reliability Block Diagram
FTA	Fault Tree Analysis
GFT	Generalized Fault Tree
HBE	Hybrid Basic Event
MCS	Monte Carlo Simulation
NPV	Net Present Value
PDAE	Partial Differential Algebraic Equation
pdf	Probability density function
PV	Photovoltaic
PVA	Present Value of Annuity
RBD	Reliability Block Diagram
RDFT	Repairable Dynamic Fault Tree
SFT	Static Fault Tree
SHA	Stochastic Hybrid Automaton
SHyFTA	Stochastic Hybrid Fault Tree Automaton
SHyFTOO	Stochastic Hybrid Fault Tree Object Oriented library
SPN	Stochastic Petri Nets
TDFT	Temporal Dynamic Fault Tree

Appendix A. The SHyFTOO Matlab®Library

The SHyFTOO library is a software toolbox for the modelling and simulation of a Stochastic Hybrid Fault Tree Automaton. It has been coded in Matlab® following an Object-Oriented paradigm in order to allow the extension and reusability of the software components. The version hereby presented supports the modelling of Repairable Dynamic Fault Trees that can be coupled with a dynamic system of the Matlab® Simulink. This represents a powerful improvement over the previous software version [22] because the modelling of dynamic systems with Simulink results of easy implementation and can further alleviate the effort of engineers and risk practitioners.

The SHyFTOO library is made up of several software components. In order to codify a SHyFTA model, the following files need to be edited:

- SHyFTAmain.m: the main script of a SHyFTA model;
- initFaultTree.m: the structure of the fault tree model;
- SHyFTA.slx: the Simulink model of the dynamic system.
- shyftaMetrics.m: this is used to update the SHyFTA variables defined in the SHyFTAmain script.

Appendix A.1. Script SHyFTAmain

The SHyFTAmain script contains the main global variables of the simulation. The file is constituted by several sections and only the ones identified by a comment %%TO EDIT %% need to be modified. Among them, it is possible to recognize the *SIMULATION PARAMETERS* and the *SHYFTA VARIABLES*.

The section *SIMULATION PARAMETERS* contains the global variables of the simulation:

- *debugMode* (boolean): this flag enables the debug mode with a log file that reports all the events occurring in a simulation;
- *iter* (integer): this variable represents the maximum number of iterations of the Monte Carlo simulation;
- *errorTolerance* (float): this variable defines the accepted width around the mean value;
- *confidenceLevel* (< 1, float): this variable represents the confidence level required to stop the simulation;
- *TimeStep* (float): it represents the time-step of the simulation. The default value is 1;
- *randomSeed* (array of integer): this array contains a bunch of random seeds to use in the "*Uniform Random Number"* block generator of Simulink.

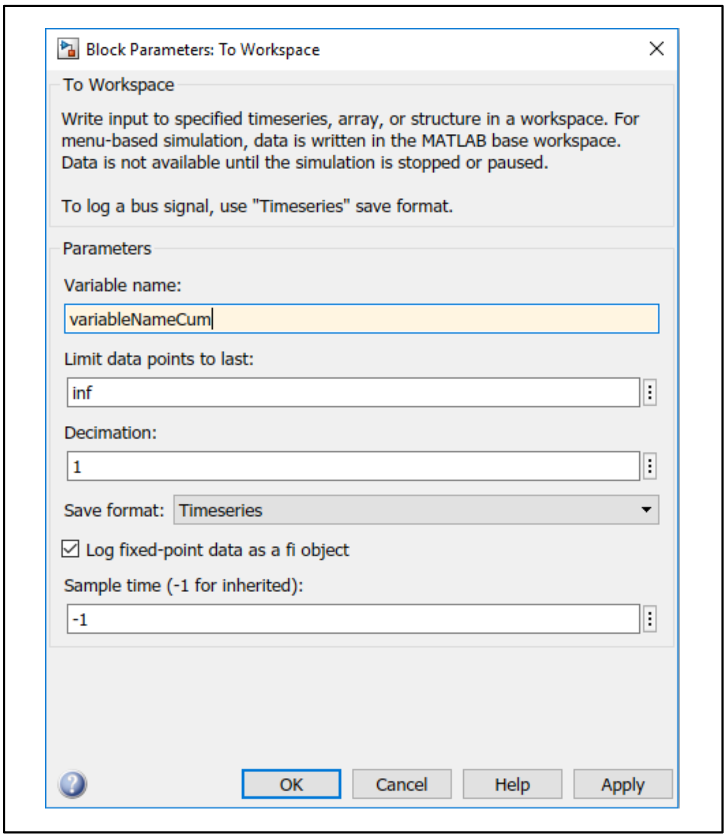
The section *SHYFTA VARIABLES* must include, if needed, the definition of specific user variables. Among them, it is often common to include the continuous variables of the physical process.

Moreover, in order to keep track of the evolution of these variables, a dedicated Simulink block *ToWorkspace* has to be used in the Simulink model SHyFTA.slx, setting, as shown in Figure A1, the parameter *VariableName* = *variableNameCum* and the *Save format* = *Timeseries*.

Appendix A.2. Script Initfaulttree

The initFaultTree script contains the structure and the parameters of the fault tree. In order to write correctly the fault tree, the variables have to be defined before being used and recalled within the script. Therefore, it is strongly suggested to initialize all the basic events before and, afterwards, all the gates.

With the SHyFTOO library, the basic event have to be initialized according to the following syntax: beName = BasicEvent (string BasicEventName, string pdfFailure, string pdfRepair, float [] paramsFailure, float [] paramsRepair, string pdfFailureSpare*, float [] paramFailureSpare*), where the symbol * indicates optional inputs and they are used if the basic event is a spare component of a Spare gate.



To Workspace

Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.

To log a bus signal, use "Timeseries" save format.

Parameters

Variable name: variableNameCum

Limit data points to last: inf

Decimation: 1

Save format: Timeseries

☑ Log fixed-point data as a fi object

Sample time (-1 for inherited): -1

Figure A1. Customization of a Simulink block of type ToWorkspace.

The SHyFTOO supports the native probability density functions of Matlab® [62]:

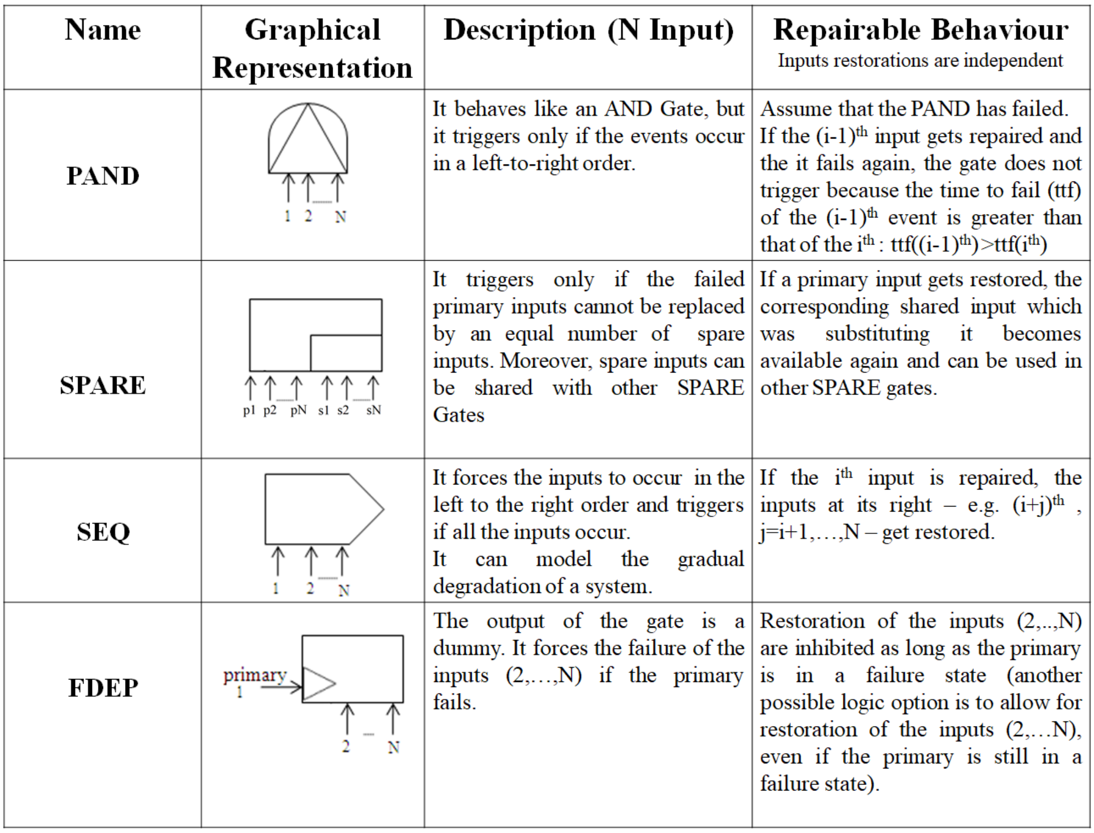
- Exponential ('exp'): in this case the array *paramsFailure* (*paramsRepair)* contains a single value (e.g., failure/repair rate);
- Weibull ('wei'): in this case the array *paramsFailure* (*paramsRepair)* contains three values (e.g., shape, scale and location parameter);
- Beta ('beta'): in this case the array *paramsFailure* (*paramsRepair)* contains two parameters;
- for the other probability density function, the Matlab® documentation can be consulted [49];
- Hybrid ('hybrid'): finally, in case of hybrid basic events, the array *paramsFailure* (*paramsRepair)* must be empty (e.g., paramsRepair = []).

Once the basic events of the fault tree are defined, it is possible to implement the intermediate and the top event gates, according to the following syntax: gateName = Gate (*string GateName, string GateType, boolean IsFailureGate, BasicEvent* [] *inputs, BasicEvent* [] *spares**), where the symbol * indicates optional inputs and they are used if the gate is of Spare type. The list of allowed GateType is the following [22] and a thorough description is provided in Figure A2.

- AND ('AND'): the boolean AND gate. It fails as soon all the inputs have failed;
- OR ('OR'): the boolean OR gate. It fails as soon one input has failed;
- VOTING ('VKx'): the voting gate triggers as soon x input out of N have failed. The value of x has to be modified accordingly (e.g., a voting gate 3 out of N will be defined as 'VK3');
- PAND ('PAND'): the PAND gate is the dynamic gate that triggers if the inputs fail in the left-to-right order;
- SPARE ('SPARE'): the SPARE gate is the dynamic gate that triggers if the primary inputs are failed and no spare inputs are available to replace them;

- SEQ (‘SEQ’): the SEQ gate is a dynamic gate that forces the input at failing from the right to the left. It triggers as soon the last input has failed;
- FDEP (‘FDEP’): the FDEP gate is a dynamic gate that imposes the failure of the inputs connected to the gate on the basis of the state of its primary input.

The parameter *IsFailureGate* is a boolean value that allows to specify if the gate can be restored to the good status if the inputs are repairable [21]. As for the inputs of the gate, the SHyFTOO allows the codification of extended fault tree.



Name	Graphical Representation	Description (N Input)	Repairable Behaviour Inputs restorations are independent
PAND	1 2 N	It behaves like an AND Gate, but it triggers only if the events occur in a left-to-right order.	Assume that the PAND has failed. If the (i-1)th input gets repaired and the it fails again, the gate does not trigger because the time to fail (ttf) of the (i-1)th event is greater than that of the ith : ttf((i-1)th)>ttf(ith)
SPARE	p1 p2 pN s1 s2 sN	It triggers only if the failed primary inputs cannot be replaced by an equal number of spare inputs. Moreover, spare inputs can be shared with other SPARE Gates	If a primary input gets restored, the corresponding shared input which was substituting it becomes available again and can be used in other SPARE gates.
SEQ	1 2 N	It forces the inputs to occur in the left to the right order and triggers if all the inputs occur. It can model the gradual degradation of a system.	If the ith input is repaired, the inputs at its right – e.g. (i+j)th , j=i+1,…,N – get restored.
FDEP	primary 1 2 N	The output of the gate is a dummy. It forces the failure of the inputs (2,…,N) if the primary fails.	Restoration of the inputs (2,..,N) are inhibited as long as the primary is in a failure state (another possible logic option is to allow for restoration of the inputs (2,…N), even if the primary is still in a failure state).

Figure A2. Dynamic Fault Tree gate behaviour.

These fault trees support the cascade of dynamic gates, (i.e., the inputs of the dynamic gates can be of basic events or gates) [36].

Appendix A.3. Simulink Model SHyFTA(.slx) and SHYFTA_TEMPLATE(.slx)

The Simulink model SHyFTA(.slx) is the dynamic system that describes the physical process of the SHyFTA model. This Simulink model must include some relevant Simulink blocks (collected in a library of the file SHyFTA_TEMPLATE.slx of the SHyFTOO library package, Figure A3).

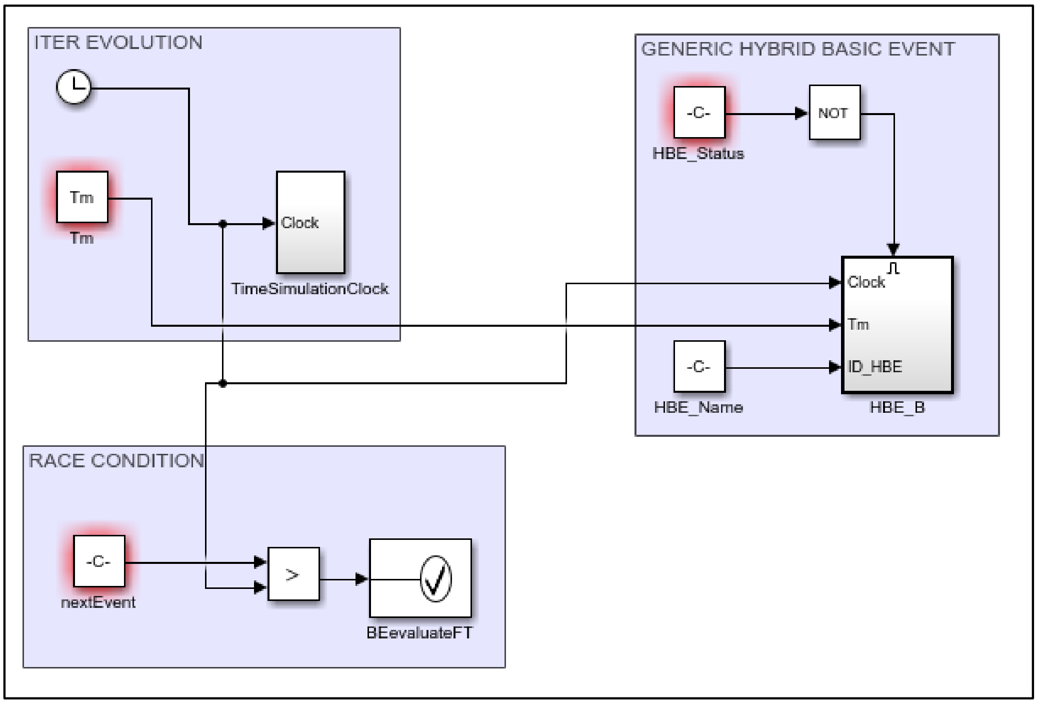


Figure A3. Simulink blocks of the SHyFTA_TEMPLATE.slx file, to use in the Simulink model of a SHyFTA.

These relevant blocks allow the coupling with the stochastic model (the fault tree components defined in the script initFaultTree) and they have to be used as explained in the following:

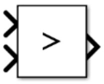
- The "*ITER EVOLUTION*" must copied as it is and must not be modified. It is an ensemble of blocks that control, for each iteration, the simulation time. When the mission time *Tm* has reached, a new iteration is automatically restarted. The marking of the "*ITER EVOLUTION*" components are summarized in Table A1.

Table A1. Marking of the components of the ITER EVOLUTION Simulink block.

Element	Description
Mission Time T_m	It represents the mission time of the system. This parameter is set in the script SHyFTAmain.m The Simulink block must not be deleted or modified.
Clock	It is the Simulink block representing the simulation clock. The Simulink block must not be deleted.
TimeSimulationClock	This block evaluates the time evolution until the end of the mission time. The Simulink block must not be deleted or modified.

- The "*RACE CONDITION*" must be copied as it is and must not be modified. The marking of the "*RACE CONDITION*" components are summarized in Table A2. This ensemble controls the occurrence of the next discrete event time-point of status change for the basic events of the fault tree. If the *nextEvent* time-point is higher than the current clock, the iteration is paused, and the fault tree status is evaluated. This block is in race condition against all the "*GENERIC HYBRID BASIC EVENT*" ensembles, as shown in Figure A5.

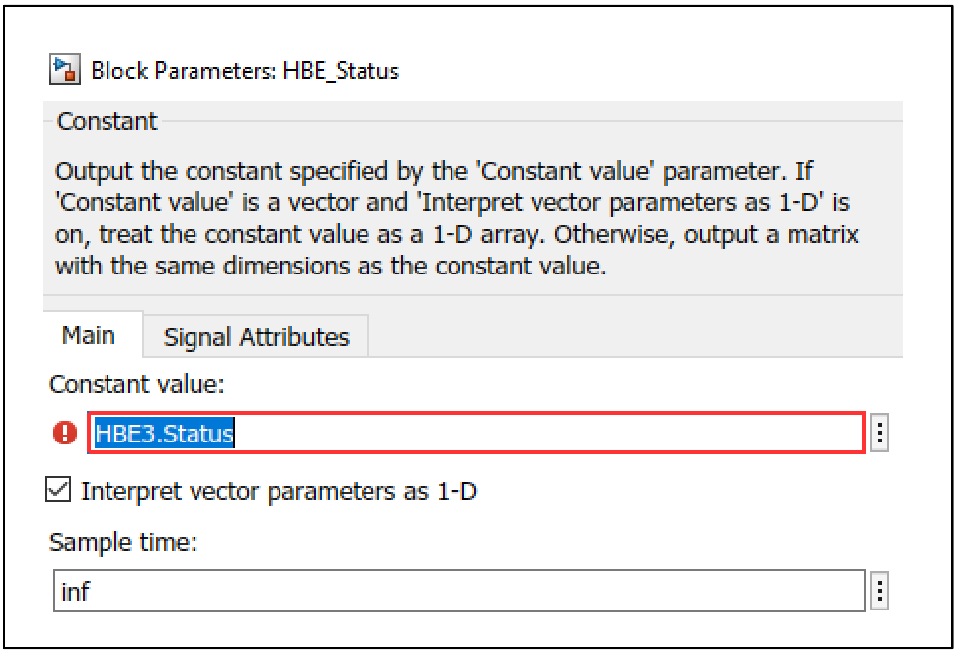
Table A2. Marking of the components of the RACE CONDITION Simulink block.

Element	Description
nextEvent	It contains a global variable array that keeps track of the events (of the basic events) that must occur. The Simulink block must not be deleted or modified.
Relation Operator 	It is the Simulink block that compares two inputs. The output is a Boolean value (true if the input 1 is greater than input 2, false vice versa). The Simulink block must not be deleted.
BEevaluteFT	This block is a Simulink Assertion that evaluates the occurrence of the next basic event in race condition with all the existing hybrid basic events handled in the GENERIC HYBRID BASIC EVENT blocks. The Simulink block must not be deleted or modified.

- For each hybrid basic event of the fault tree, the Simulink model SHyFTA(.slx) has to contain a block ensemble of type "*GENERIC HYBRID BASIC EVENT*". The task of this block is to verify the status of the generic hybrid basic event that is in race condition against all the other hybrid basic events and the regular basic events. The race condition is handled in the "*RACE CONDITION*" ensemble.

Conversely from the previous blocks, for each "*GENERIC HYBRID BASIC EVENT*", the setting of the *HBE_Status* and *HBE* blocks must be modified. For instance, as shown in Figure A4, these settings have to specify the name of the hybrid basic event, as defined in the initFaultTree script. In Figure A4 it is possible to see that the *Constant value* takes the value *BasicEventName.Status* (e.g., in that case the name of the hybrid basic event specified in the initFaultTree is HBE3). The same reasoning applies for the block HBE.

The marking of the "*GENERIC HYBRID BASIC EVENT*" components are summarized in Table A3.



Block Parameters: HBE_Status

Constant

Output the constant specified by the 'Constant value' parameter. If 'Constant value' is a vector and 'Interpret vector parameters as 1-D' is on, treat the constant value as a 1-D array. Otherwise, output a matrix with the same dimensions as the constant value.

Constant value:

HBE3.Status

☑ Interpret vector parameters as 1-D

Sample time:

inf

Figure A4. Customization of a Simulink block of type Constant.

Table A3. Marking of the components of the GENERIC HYBRID BASIC EVENT Simulink block.

Element	Description
HBE_Status	It contains the SHyFTOO variable of the status of the hybrid basic event of the corresponding "GENERIC HYBRID BASIC EVENT" block. The value of the HBE_Status of each GENERIC HYBRID BASIC EVENT" block must be modified.
Logical Port NOT	It is the Simulink block that invert the logical input. The output is a Boolean value (true if the input is false and false if the input is true). The Simulink block must not be deleted.
HBE_Name	It contains the SHyFTOO variable of the name of the hybrid basic event of the corresponding "GENERIC HYBRID BASIC EVENT" block. The value of the HBE_Name of each GENERIC HYBRID BASIC EVENT" block must be modified.

The *HBE_B* block of the "*GENERIC HYBRID BASIC EVENT*" shown in Figure A5 controls the next time-event of the corresponding hybrid basic event. The marking of the components of this block are described in Table A4.

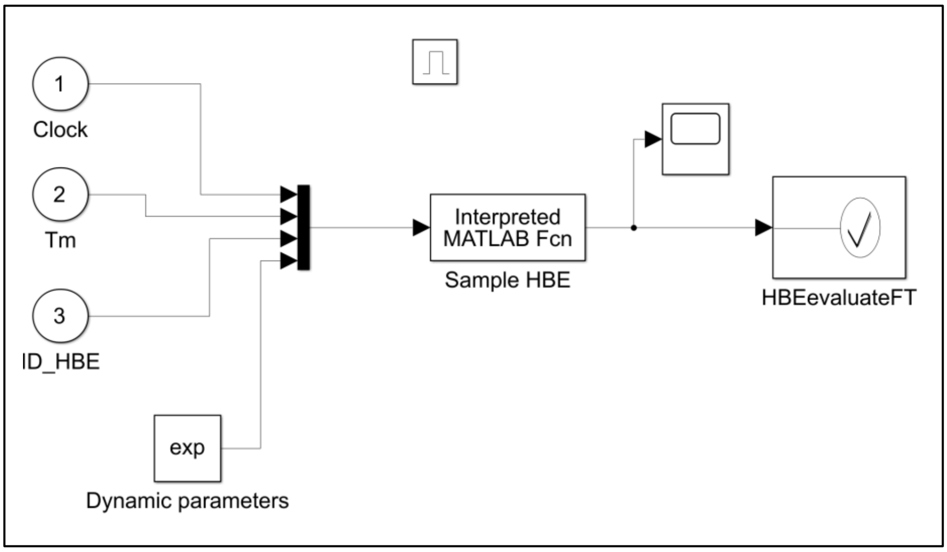
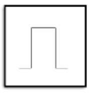
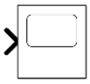


Figure A5. The Simulink implementation of the "Generic Hybrid Basic Event" block.

The physical process is the most suitable simulation scope for the hybrid events that are characterized by dynamic parameters that change throughout the evolution of the physical process. This block contains a Matlab® function (*Dynamic parameters*) that updates the failure parameters of the component at each time-step of the iteration and pass it, together with other inputs (e.g., clock, mission time *Tm*, and id of the HBE) to an "*Interpreted Matlab Function*", named "*Sample HBE*", that samples whether the hybrid basic event has occurred or not. In fact, at any timestep, the "*Interpreted Matlab Function*" inverts the probability density function characterizing the hybrid basic event and compares it with a real random value in (0, 1]. If the comparison is positive, an exception is raised and handled in the *HBEevaluateFT* assertion block. In this case, the Simulink iteration is paused, and the control is passed to the SHyFTOO library to verify the fault tree status. Otherwise, the Simulink iteration continues normally. The mechanism hereby described is the one that allows the race condition together with the block "*RACE CONDITION*".

Table A4. Marking of the components of the HBE Simulink block.

Element	Description
Clock	It takes the input of the Simulink clock (refer to Table A1). This input must not be deleted or modified.
T_m	It takes the input of the mission time (refer to Table A1). This input must not be deleted or modified.
ID_HBE	It takes the input of the HBE_Name (refer to Table A3). This input must not be deleted or modified.
Dynamic parameters	Additional input block that can vary (depends on the physical process). The input must be modified according to the dependencies between dynamic failure parameters and the physical process.
SAMPLE_HBE	It is a function that samples the hybrid basic event occurrence. This input must not be deleted or modified.
Enable/Disable 	It is the Simulink block that enables or disables the block. It is used to improve the performance of the software. The Simulink block must not be deleted.
Scope 	It is the Simulink block that plot the input variable. The Simulink block can be deleted.
HBEevaluateFT	This block evaluates the occurrence of the hybrid basic event in race condition with all the other existing hybrid basic events handled in the other GENERIC HYBRID BASIC EVENT blocks and with the regular basic events, handled in the block RACE CONDITION. The Simulink block must not be deleted or modified.

Appendix A.4. Script shyftaMetrics

The script shyftaMetrics must be modified on the basis of the custom user variables that have been defined in the main script SHyFTAmain and that need to be updated at the end of each iteration. This can be, for example, the case of the continuous variables of the physical process that are averaged at the end of the simulation. The script in the Algorithm A1 applies for the generic variable name variableName defined in the SHyFTAmain script. For each custom variable, the lines from 2 to 8 must be copied and pasted, modifying the variable of interest.

Algorithm A1. Code for the definition of a custom variable with the SHyFTOO library.

```
1:    t2fill = [0:Tm];
2:    observedTimes = variableName.Time';
3:    observedData = variableName.Data';
4:    [~,ind] = ismember(t2fill,observedTimes);
5:    inputData = [0, observedData];
6:    ind = ind +1;
7:    corrData = inputData(ind)';
8:    variableNameCum = variableNameCum +corrData;
```

As already said, in order to keep track of the evolution of these variables, a dedicated Simulink block *ToWorkspace* has to be used in the Simulink model SHyFTA.slx, setting, as shown in Figure A5, the parameter *VariableName* = *variableNameCum* and the *Save format* = *Timeseries*.

Appendix A.5. Properties of the Basic Events and Gates

Tables A5 and A6 contain the list of attributes (or properties) of the SHyFTOO library defined for the basic events and gates. They can be easily accessed in the Matlab® workspace using the syntax *Component.Property* (e.g., BE3.Name).

Table A5. Attributes (or Properties) of the basic events.

Name	Description
Name	Name of the component
Index	Unique identifier in the Fault Tree
InputOf	List of the parent components
SpareOf	List of parent components (in case the component is a spare element)
InUseBy	Index of the spare gate in which the component is in use
Status	State of the component
FailureTime	Time of next failure
RepairTime	Time of next repair
UpTime	Cumulative time in GOOD status
NFailure	Incremental counter of failure occurrence
TimeOfFailureArray	Array containing the time of last failure

Table A6. Attributes of the Gates.

Name	Description
Type	Define the type of the gate
Inputs	List of inputs
Spares	List of spares (in case of Spare Gate)
IsFailureGate	Define if the gate is a failure gate

Appendix B.

This section contains information about the Simulink models of the case study discussed in Section 3. They have been organized in different Tables A7–A10 that refer respectively to the manuscript Figures 4–6 and Figure 8.

Table A7. Marking of the components of the hybrid basic events HBE1 and HBE5 (refer to Figure 4).

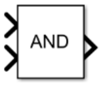
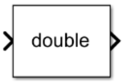
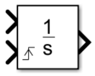
Element	Description
Clock	It takes the input of the Simulink clock of Table A1.
Mission Time	It takes the input of the mission time of Table A1.
TOP_EVENT_STATUS	It is the signal of Top Event status (Top.Status).
HBE1_Status	Refer to the HBE_Status (refer to Table A3). The block must be set = HBE1.Status (or HBE5.Status)
HBE1	Refer to the HBE_Name (refer to Table A3). The block must be set = 'HBE1' (or 'HBE5')
NOT	Refer to the Logical Port "NOT" (refer to Table A3).
Logical Port 	It is the Simulink block "AND" gate. It returns a Boolean value (true if both the inputs are true, false in all the other cases). The output of this block allows to integrate the aging of the component represented by the hybrid basic event. The aging increases only if the component is good and the system is available (if the TOP EVENT STATUS is false the component is supposed not to work).
Data type conversion 	It is the Simulink block that converts its input in a double type variable.
Integrator 	It is the Simulink block integrator. It returns the aging of the component. Input 1 is the on/off of the component Input 2 is the reset condition corresponding with the status of the HBE1_Status of the component. In fact, it is assumed that the components aging is reset to zero when the component is restored after a fault.

Table A8. Marking of the components of the hybrid basic events HBE2 (referring to Figure 5).

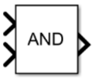
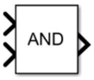
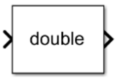
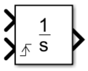
Element	Description
Clock	It takes the input of the Simulink clock (refer to Table A1).
Mission Time	It takes the input of the mission time (refer to Table A1).
TOP_EVENT_STATUS	It is the signal of Top Event status (Top.Status).
HBE2_Status	Refer to the HBE_Status (refer to Table A3). The block must be set = HBE2.Status
HBE2_InUseBy	It contains the SHyFTOO variable of the property "InUseBy" of the hybrid basic event HBE2. The block must be set = HBE2.InUseBy
HBE2	Refer to the HBE_Name (refer to Table A3). The block must be set = 'HBE2'
NOT	Refer to the Logical Port "NOT" (refer to Table A3).
Logical Port (left AND) 	It is the Simulink block "AND" gate. It returns a Boolean value (true if both the inputs are true, false in all the other cases). The output of this block returns the working (on/off) condition of the standby component. It is true if the component status is OK and if the standby condition is not false.
Logical Port (right AND) 	It is the Simulink block "AND" gate. It returns a Boolean value (true if both the inputs are true, false in all the other cases). The output of this block allows to integrate the aging of the component represented by the hybrid basic event. The aging increases only if the component is good (and not in standby) and the system is available (if the TOP EVENT STATUS is false the component is supposed not to work).
Data type conversion 	It is the Simulink block that converts its input in a double type variable.
Integrator 	It is the Simulink block integrator. It returns the aging of the component. Input 1 is the on/off of the component. Input 2 is the reset condition corresponding with the status of the HBE2_Status of the component. It is assumed that the components aging is reset to zero when the component is restored after a fault.

Table A9. Marking of the components of the hybrid basic events HBE6 (referring to Figure 6).

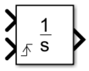
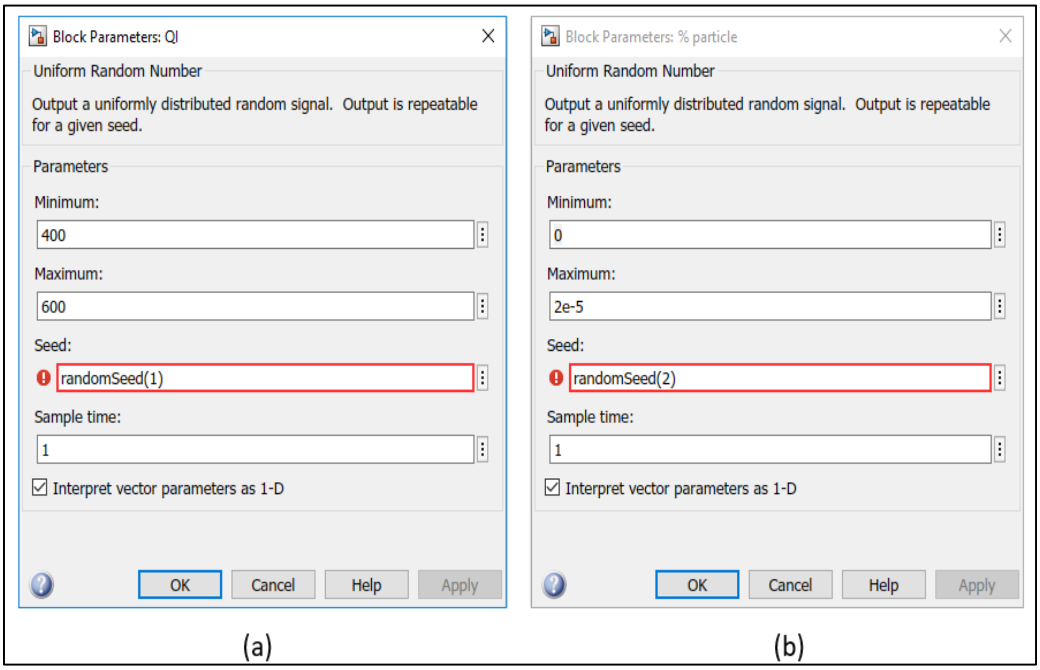
Element	Description
% Particle	It is a Simulink "uniform random number" generator that models the percentage of solid particles of the mixture. The block is set as shown in Figure A6b
QI	It is a Simulink "uniform random number" generator that models the instantaneous volume (flow rate) of mixture. The block is set as shown in Figure A6a
TOP_EVENT_STATUS	It is the signal of Top Event status (Top.Status)
¬TOP_EVENT_STATUS	It is the neglection of Top Event status (Top.Status).
HBE2_Status	Refer to the HBE_Status (refer to Table A3). The block must be set = HBE6.Status
HBE6	Refer to the HBE_Name (refer to Table A3). The block must be set = 'HBE6'
Product	It is the Simulink block of the product operation. It returns the quantity of solid particle in the instantaneous infinitesimal volume of mixture (if the Top Event status is bad, the process is stopped and the product is null). The output is the input of the integrator block.
Integrator 	It is the Simulink block integrator. It returns the volume of the solid particle deposited in the solid tank. This output is the input of the HBE6_SAMPLING block that verifies when the solid tank threshold is reached (event occurrence of the hybrid basic event HBE6). Input 1 is the output of the product block above described. Input 2 is the reset condition, represented by the status of the top event. When the top event occurs, the process is stopped (and the solid tank is emptied).

Figure A6 shows the setting of the block "QI" and "% Particle" according to the characteristics of the physical process described in the case study.



(a)

Block Parameters: QI

Uniform Random Number

Output a uniformly distributed random signal. Output is repeatable for a given seed.

Parameters

Minimum: 400

Maximum: 600

Seed: randomSeed(1)

Sample time: 1

☑ Interpret vector parameters as 1-D

(b)

Block Parameters: % particle

Uniform Random Number

Output a uniformly distributed random signal. Output is repeatable for a given seed.

Parameters

Minimum: 0

Maximum: 2e-5

Seed: randomSeed(2)

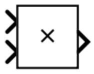
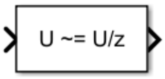
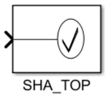
Sample time: 1

☑ Interpret vector parameters as 1-D

Figure A6. Block Parameters for the variables (**a**) QI and (**b**) % particle.

In order to avoid the repetition of iterations with the same evolution (for the correct implementation of a Monte Carlo simulation), the *Seed* parameter must be set with a random number. This feature can be obtained by using a global array variable *randomSeed* that is initialized at the beginning of each new iteration (as previously described in the script <<*SHyFTAmain.m*>>).

Table A10. Marking of the components of the physical process (referring to Figure 8).

Element	Description
QI	It is the signal of QI (refer to Table A9)
TOP_EVENT_STATUS	It is the signal of Top Event status (Top.Status) in the same block
⌐TOP_EVENT_STATUS	It is the neglection of Top Event status (Top.Status) in the same block
Product (at the top) 	It is the Simulink block of the product operation. It returns the instantaneous flow QI processed by the distillation column when the system is working (if the Top Event status is bad, the process is stopped and the product is null). The output is the input of the "To Workspace" block QI_Mixture.
QI_Mixture	It is the Matlab "To Workspace" block to store the simulation variable. It is used to store the instantaneous flow QI processed when the system is working.
Product (at the bottom)	It is the Simulink block of the product operation. It returns the instantaneous flow QI lost when the system is not working (if the Top Event status is bad, the variable ⌐TOP_EVENT_STATUS is true and the product is non null). The output is the input of the "To Workspace" block QI_Lost.
QI_Lost	It is the Matlab "To Workspace" block to store the simulation variable. It is used to store the instantaneous flow QI lost when the system is stopped (or not working).
Detect Change 	It is the Simulink block "Detect Change". It triggers a Boolean true value as soon its input changes. It is used to activate the SHA_TOP assertion The input is the top event status.
Assertion Block 	It is the Simulink "Assertion Block". It recalls the script SHA_TOP of Algorithm 3 (discussed in Section 3.2.4). The input is the neglection of the Detect Change block.

References

1. Avizienis, A.; Laprie, J.C.; Randell, B.; Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.* **2004**, *1*, 11–33. [CrossRef]
2. Aizpurua, J.I.; Muxika, E. Model-based design of dependable systems: Limitations and evolution of analysis and verification approaches. *Int. J. Adv. Secur.* **2013**, *6*, 12–31.
3. Labeau, P.E.; Smidts, C.; Swaminathan, S. Dynamic reliability: Towards an integrated platform for probabilistic risk assessment. *Reliab. Eng. Syst. Saf.* **2000**, *68*, 219–254. [CrossRef]
4. Raoni, R.; Secchi, A.R. Procedures to Model and Solve Probabilistic Dynamic System Problems. *Reliab. Eng. Syst. Saf.* **2019**, *191*, 106554. [CrossRef]
5. Di Stefano, S. Dependability assessment of critical systems. *J. Ambient Intell. Hum. Comput.* **2015**, *6*, 713–720. [CrossRef]
6. Zhang, H.; Saporta, B.; Dufoura, F.; Deleuzed, G. Dynamic reliability by using simulink and stateflow. *Chem. Eng. Trans.* **2013**, *33*, 529–534.
7. Dugan, J.B.; Venkataraman, B.; Gulati, R. DIFTree: A software package for the analysis of dynamic fault tree models. In Proceedings of the IEEE Annual Reliability and Maintainability Symposium, Philadelphia, PA, USA, 13–16 January 1997.
8. Sullivan, K.J.; Dugan, J.B.; Coppit, D. The Galileo fault tree analysis tool. In Proceedings of the IEEE Digest of Papers, Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing (Cat. No. 99CB36352), Madison, WI, USA, 15–18 June 1999; pp. 232–235.
9. Ramani, S.; Gokhale, S.S.; Trivedi, K.S. SREPT: Software reliability estimation and prediction tool. *Perform. Eval.* **2000**, *39*, 37–60. [CrossRef]
10. Montani, S.; Portinale, L.; Bobbio, A.; Codetta-Raiteri, D. Radyban: A tool for reliability analysis of dynamic fault trees through conversion into dynamic Bayesian networks. *Reliab. Eng. Syst. Saf.* **2008**, *93*, 922–932. [CrossRef]
11. Patelli, E.; Tolo, S.; George-Williams, H.; Sadeghi, J.; Rocchetta, R.; de Angelis, M.; Broggi, M. OpenCossan 2.0: An efficient computational toolbox for risk, reliability and resilience analysis. In Proceedings of the Joint ICVRAM ISUMA UNCERTAINTIES Conference, Florianópolis, Brazil, 8–11 April 2018.
12. Pérez Castañeda, G.A.; Aubry, J.-F.; Brinzei, N. DyRelA (dynamic reliability and assessment). In Proceedings of the First Workshop on DYnamic Aspects in DEpendability Models for Fault-Tolerant Systems, Valencia, Spain, 27 April 2010.
13. Arnold, F.; Belinfante, A.; Van der Berg, F.; Guck, D.; Stoelinga, M. DFTCalc: A tool for efficient fault tree analysis. In Proceedings of the International Conference on Computer Safety, Reliability, and Security, Toulouse, France, 24–27 September 2013.
14. Boudali, H.; Nijmeijer, A.; Stoelinga, M.I. DFTSim: A simulation tool for extended dynamic fault trees. In Proceedings of the 2009 Spring Simulation Multiconference, San Diego, CA, USA, 22–27 March 2009.
15. Boudali, H.; Crouzen, P.; Stoelinga, M. CORAL-a tool for compositional reliability and availability analysis. In Proceedings of the 19th international conference on Computer Aided Verification, Berlin, Germany, 3–7 July 2007.
16. Batteux, M.; Prosvirnova, T.; Rauzy, A.; Yang, L. Reliability assessment of phased-mission systems with AltaRica 3.0. In Proceedings of the 3rd International Conference on System Reliability and Safety (ICSRS 2018), Barcelona, Spain, 23–25 November 2018.
17. Chraibi, H. Dynamic reliability modeling and assessment with PyCATSHOO: Application to a test case. In Proceedings of the Probabilistic Safety Assessment and Management (PSAM), Prague, Czech Republic, 17–20 June 2013.
18. Cojazzi, G. The DYLAM approach for the dynamic reliability analysis of systems. *Reliab. Eng. Syst. Saf.* **1996**, *52*, 279–296. [CrossRef]
19. Courtney, T.; Gaonkar, S.; Keefe, K.; Rozier, E.W.; Sanders, W.H. Möbius 2.3: An extensible tool for dependability, security, and performance evaluation of large and complex system models. In Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems & Networks, Lisbon, Portugal, 29 June–2 July 2009.

20. Manno, G.; Chiacchio, F.; Compagno, L.; D'Urso, D.; Trapani, N. MatCarloRe: An integrated FT and Monte Carlo Simulink tool for the reliability assessment of dynamic fault tree. *Expert Syst. Appl.* **2012**, *39*, 10334–10342. [CrossRef]
21. Manno, G.; Chiacchio, F.; Compagno, L.; D'Urso, D.; Trapani, N. Conception of Repairable Dynamic Fault Trees and resolution by the use of RAATSS, a Matlab® toolbox based on the ATS formalism. *Reliab. Eng. Syst. Saf.* **2014**, *121*, 250–262. [CrossRef]
22. Chiacchio, F.; Aizpurua, J.I.; Compagno, L.; D'Urs, D. SHyFTOO, an object-oriented Monte Carlo simulation library for the modelling of Stochastic Hybrid Fault Tree Automaton. *Expert Syst. Appl.* **2019**. submitted.
23. Ericson, C.A. Fault Tree Analysis—A History. In Proceedings of the 17th International System Safety Conference, Orlando, FL, USA, 16–21 August 1999; pp. 1–9.
24. Vesely, W.E.; Goldberg, F.F.; Roberts, N.H.; Haasl, D.F. *Fault Tree Handbook (No. NUREG-0492)*; Nuclear Regulatory Commission: Washington, DC, USA, 1981.
25. Jung, W.S.; Han, S.H.; Ha, J. A fast BDD algorithm for large coherent fault trees analysis. *Reliab. Eng. Syst. Saf.* **2004**, *83*, 369–374. [CrossRef]
26. Zang, X.; Sun, N.; Trivedi, K.S. A BDD-based algorithm for reliability analysis of phased-mission systems. *IEEE Trans. Reliab.* **1999**, *48*, 50–60. [CrossRef]
27. Schneeweiss, W.G. Fault-tree analysis using a binary decision tree. *IEEE Trans. Reliab.* **1985**, *34*, 453–457. [CrossRef]
28. Locks, M.O. Recursive disjoint products: A review of three algorithms. *IEEE Trans. Reliab.* **1982**, *31*, 33–35. [CrossRef]
29. Aghaie, A. Evaluating ISO 9001: 2000 implementation using fault tree analysis (FTA). *Total Qual. Manag. Bus. Excell.* **2004**, *15*, 971–983. [CrossRef]
30. Contini, S.; Cojazzi, G.G.M.; De Cola, G. On the exact analysis of non-coherent fault trees: The ASTRA package (PSAM-0285). In Proceedings of the Eighth International Conference on Probabilistic Safety Assessment & Management (PSAM), New Orleans, LA, USA, 14–18 August 2006.
31. Chen, P.; Mou, J.; Yajun, L. Risk analysis of maritime accidents in an estuary: A case study of Shenzhen Waters. *Sci. J. Marit. Univ. Szczec.* **2015**, *42*, 54–62.
32. Chybowski, L. Assessment of Reliability and Availability of Fishing Vessels Power, Propulsion and Technological Plants Based. *Stud* **2009**, *18*, 39–44.
33. Misra, K.B. *Handbook of Performability Engineering*; Springer Science & Business Media: Berlin, Germany, 2008.
34. Čepin, M.; Mavko, B. A dynamic fault tree. *Reliab. Eng. Syst. Saf.* **2002**, *75*, 83–91. [CrossRef]
35. Ammar, M.; Hamad, G.B.; Mohamed, O.A.; Savaria, Y. Towards an Accurate Probabilistic Modeling and Statistical Analysis of Temporal Faults via Temporal Dynamic Fault-Trees (TDFTs). *IEEE Access* **2019**, *7*, 29264–29276. [CrossRef]
36. Codetta-Raiteri, D. Integrating several formalisms in order to increase Fault Trees' modeling power. *Reliab. Eng. Syst. Saf.* **2011**, *96*, 534–544. [CrossRef]
37. Codetta-Raiteri, D.; Franceschinis, G.; Iacono, M.; Vittorini, V. Repairable fault tree for the automatic evaluation of repair policies. In Proceedings of the International Conference on Dependable Systems and Networks, Florence, Italy, 28 June–1 July 2004.
38. Bouissou, M. A generalization of dynamic fault trees through Boolean logic driven Markov processes (BDMP)®. In Proceedings of the16th European Safety and Reliability Conference (ESREL'07), Stavanger, Norway, 25–27 June 2007.
39. Yuge, T.; Tamura, N.; Yanagi, S. Repairable Fault Tree Analysis Using Renewal Intensities. *Qual. Technol. Quant. Manag.* **2012**, *9*, 231–241. [CrossRef]
40. Aizpurua, J.I.; Catterson, V.M.; Papadopoulos, Y.; Chiacchio, F.; Manno, G. Improved dynamic dependability assessment through integration with prognostics. *IEEE Trans. Reliab.* **2017**, *66*, 893–913. [CrossRef]
41. Li, Y.F.; Huang, H.Z.; Yu, L.; Xiao Haiqing, L.N.C. A new fault tree analysis method: Fuzzy dynamic fault tree analysis. *Maint. Reliab.* **2012**, *14*, 208–214.
42. Chybowski, L.; Gawdzińska, K.; Wiśnicki, B. Qualitative Importance Measures of Systems Components—A New Approach and Its Applications. *Manag. Syst. Prod. Eng.* **2018**, *24*, 237–246. [CrossRef]
43. Merle, G.; Roussel, J.-M.; Lesage, J.-J. Dynamic fault tree analysis based on the structure function. In Proceedings of the Annual Reliability and Maintainability Symposium, Lake Buena Vista, FL, USA, 24–27 January 2011.

44. Merle, G.; Roussel, J.-M.; Lesage, J.-J.; Bobbio, A. Probabilistic algebraic analysis of fault trees with priority dynamic gates and repeated events. *IEEE Trans. Reliab.* **2010**, *59*, 250–261. [CrossRef]
45. Rauzy, A.B. Sequence algebra, sequence decision diagrams and dynamic fault trees. *Reliab. Eng. Syst. Saf.* **2011**, *96*, 785–792. [CrossRef]
46. Aslansefat, K.; Latif-Shabgahi, G.R. A Hierarchical Approach for Dynamic Fault Trees Solution through Semi-Markov Process. *IEEE Trans. Reliab.* **2019**. [CrossRef]
47. Rao, K.D.; Gopika, V.; Rao, V.S.; Kushwaha, H.; Verma, A.K.; Srividya, A. Dynamic fault tree analysis using Monte Carlo simulation in probabilistic safety assessment. *Reliab. Eng. Syst. Saf.* **2009**, *94*, 872–883.
48. Ruijters, E.; Reijsbergen, D.; de Boer, P.T.; Stoelinga, M. Rare event simulation for dynamic fault trees. *Reliab. Eng. Syst. Saf.* **2019**, *186*, 220–231. [CrossRef]
49. Babykina, G.; Brinzei, N.; Aubry, J.-F.; Deleuze, G. Modeling and simulation of a controlled steam generator in the context of dynamic reliability using a Stochastic Hybrid Automaton. *Reliab. Eng. Syst. Saf.* **2016**, *152*, 115–136. [CrossRef]
50. Fan, M.; Zeng, Z.; Zio, E.; Kang, R.; Chen, Y. A stochastic hybrid systems based framework for modeling dependent failure processes. *PLoS ONE* **2017**, *12*, e0172680. [CrossRef] [PubMed]
51. Riley, D.D.; Koutsoukos, X.; Riley, K. Simulation of Stochastic Hybrid Systems using probabilistic boundary detection and adaptive time stepping. *Simul. Model. Pract. Theory* **2010**, *18*, 1397–1411. [CrossRef]
52. Yuehua, C.; Liang, J.; Bin, J.; Ningyun, L. Useful life prediction using a stochastic hybrid automata model for an ACS multi-gyro subsystem. *J. Syst. Eng. Electron.* **2019**, *30*, 154–166.
53. Kakalis, N.M.P.; Dimopoulos, G.G.; Ovrum, E. DNV COSSMOS: Complex Ship Systems Modelling and Simulation. *Det. Norske Veritas*. Available online: www.dnv.com/binaries/ship%20machinery_cossmos_tcm4-528552.pdf (accessed on 5 July 2019).
54. Manno, G.; Zymaris, A.; Kakalis, N.P.; Chiacchio, F.; Cipollone, G.; Compagno, L.; D'Urso, D.; Trapani, N. Dynamic reliability of three nonlinear aging components with different failure modes characteristics. In *Safety, Reliability and Risk Analysis, Beyond the Horizon*; Steenbergen, R.D.J.M., VanGelder, P.H.A.J.M., Miraglia, S., Vrouwenvelder, A.C.W.M., Eds.; CRC Press: Boca Raton, FR, USA, 2013.
55. Chiacchio, F.; D'Urso, D.; Compagno, L.; Pennisi, M.; Pappalardo, F.; Manno, G. SHyFTA, a Stochastic Hybrid Fault Tree Automaton for the modelling and simulation of dynamic reliability problems. *Expert Syst. Appl.* **2016**, *47*, 42–57. [CrossRef]
56. Chiacchio, F.; Iacono, A.; D'Urso, D.; Compagno, L. A general framework for dependability modelling coupling discrete-event and time-driven simulation. *Reliab. Eng. Syst. Saf.* **2019**, submitted.
57. Varghese, B.; Buyyab, R. Next generation cloud computing: New trends and research directions. *Future Gener. Comput. Syst.* **2018**, *79*, 849–861. [CrossRef]
58. MIL-HDBK-217F, Military Handbook Reliability Prediction of Electronic Equipment (1991). Available online: https://snebulos.mit.edu/projects/reference/MIL-STD/MIL-HDBK-217F-Notice2.pdf (accessed on 7 July 2019).
59. Chiacchio, F.; Famoso, F.; D'Urso, D.; Cedola, L. Performance and Economic Assessment of a Grid-Connected Photovoltaic Power Plant with a Storage System: A Comparison between the North and the South of Italy. *Energies* **2019**, *12*, 2356. [CrossRef]
60. Chiacchio, F.; Famoso, F.; D'Urso, D.; Brusca, S.; Aizpurua, J.; Cedola, L. Dynamic performance evaluation of photovoltaic power plant by stochastic hybrid fault tree automaton model. *Energies* **2018**, *11*, 306. [CrossRef]
61. Tribe, M.A.; Alpine, R.L.W. Scale economies and the "0.6 Rule". *Eng. Costs Prod. Econ.* **1986**, *10*, 271–278. [CrossRef]
62. Matlab®Online Documentation. Available online: https://it.mathworks.com/help/stats/continuous-distributions.html (accessed on 6 August 2019).