Expert Systems with Applications 39 (2012) 10334–10342

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

MatCarloRe: An integrated FT and Monte Carlo Simulink tool for the reliability assessment of dynamic fault tree

Gabriele Manno [a,*], Ferdinando Chiacchio [a], Lucio Compagno [b], Diego D'Urso [b], Natalia Trapani [b]

[a] DMI, Department of Mathematics and Informatics, University of Catania, Italy
[b] DIIM, Department of Industrial and Mechanical Engineering, University of Catania, Italy

ARTICLE INFO

Keywords:
Reliability assessment
Dynamic fault tree
Monte Carlo simulation
Continuous Time Markov Chain

ABSTRACT

With the aim of a more effective representation of reliability assessment for real industry, in the last years concepts like dynamic fault trees (DFT) have gained the interest of many researchers and engineers (dealing with problems concerning safety management, design and development of new products, decision analysis and project management, maintenance of industrial plant, etc.). With the increased computational power of modern calculators is possible to achieve results with low modeling efforts and calculating time. Supported by the strong mathematical basis of state space models, the DFT technique has increased its popularity. Nevertheless, DFT analysis of real application has been more likely based on a specific case to case resolution procedure that often requires a great effort in terms of modeling by the human operator. Moreover, limitations like the state space explosion for increasing number of components, the constrain of using exponential distribution for all kind of basic events constituting any analyzed system and the ineffectiveness of modularization for DFT which exhibit dynamic gates at top levels without incurring in calculation and methodological errors are faces of these methodologies. In this paper we present a high level modeling framework that exceeds all these limitations, based on Monte Carlo simulation. It makes use of traditional DFT systemic modeling procedure and by replicating the true casual nature of the system can produce relevant results with low effort in term of modeling and computational time. A Simulink library that integrates Monte Carlo and FT methodologies for the calculation of DFT reliability has been developed, revealing new insights about the meaning of spare gates.

1. Introduction

In recent years the importance of risk assessment in the safety context of industrial processes has increased significantly. On the one hand, companies must provide, even more than before, guarantees about the occurrence of significant risks and adopt preventive measures and mitigation of their occurrence, while one the other side, engineers need to predict the reliability of the system from the design phase, especially for critical applications. The systemic reliability representation of the plant process, the quantitative results and sensitivity analysis are straightforwardly obtained using stochastic models such as Reliability Block Diagram (RBD) and Fault Tree Analysis (FTA). These methods have gained wide acceptance for the study of reliability for many kind of plants and systems.

A fault-tree (FT) can be simply described as an analytical technique, whereby an undesirable state of the system is specified; the system is then analysed in the context of its environment and operation in order to find all credible ways in which the undesirable event can occur (Roberts, Vesely, Haasl, & Goldberg, 1981). Although there are relatively efficient algorithms for solving FTs, the main disadvantage is that dependencies of various kinds, which are in real systems, are not easily captured in the model. Indeed, traditional FT cannot capture the dynamic behavior of a system such as the sequence of events in time dependence, the replacement of spare parts and priorities of failure events (Amari, Dill, & Howald, 2003; Cepin & Mavko, 2002; Ren & Dugan, 1998; Siu, 1994).

To overcome these difficulties, dynamic-FT (DFT) was introduced, with the formalization of dynamic gates (PAND, SPARE, SEQ and FDEP). With the help of dynamic gates, the reliability behavior of systems with time dependencies can be modeled using the DFT technique, keeping the intuitive construction of traditional FT.

DFT resolution cannot rely just on Boolean algebra as well as traditional-FT because dynamic features are not captured with a binary time independent logic. Therefore, beside DFT many other methods have been devised: Dynamic Reliability Block Diagram (DRBD), Continuous Time Markov Chain (CTMC), Input/Output Interactive Markov Chain (I/O IMC), Bayesian Network, Stochastic Petri Network, etc. (Bobbio, Portinale, Minichino, & Ciancamerla, 2001; Boudali, Crouzen, & Stoelinga, 2007; Distefano et al., 2007; Dugan, Venkataraman, & Gulati, 1997; Dugan, Trivedi, Smotherman, & Geist, 1986; Montani, Portinale, Bobbio, & Codetta-Raiteri, 2008; Sullivan, Dugan, & Coppit, 1999; Volovoi, 2004).

* Corresponding author.
E-mail address: gmanno@dmi.unict.it (G. Manno).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.12.020

The complexity of modern engineering system as well as the need for realistic reliability makes their modeling and analysis a non trivial task. The analytical resolution of systems with dynamic redundancy is very difficult to accomplish. Therefore risk analysts require the use of other techniques in order to construct a comprehensive but achievable and efficient study of the system. The simulation of DFTs can help to overcome many of the difficulties raising from analytical approaches and can offer a high level modeling interface based on the FT methodology. Scenarios that may be difficult to solve analytically are easily resolved with the approach of Monte Carlo simulation. Due to the intrinsic ability to simulate the actual process and the random behavior of the system, this approach can eliminate uncertainty of the reliability modeling. It has been used for the availability, reliability and important measure estimation of complex systems (Chiacchio, Compagno, D'Urso, Manno, & Trapani, 2011; Durga Rao et al., 2009; Marseguerra & Zio, 2004; Marsaguerra, Zio, Devooght, & Labeau, 1998; Marquez, Heguedas, & Iung, 2005; Zio, Podofillini, & Levitin, 2004; Zio, Marella, & Podollini, 2007).

In this paper we present a novel approach to conduct Monte Carlo simulation for the reliability evaluation of DFTs. In our approach Monte Carlo and FT methodologies are integrated in order to allow the user to benefit of an high level *FT-like* interface that exploits the power of Monte Carlo simulation to overcome many of the difficulties raising from the use of analytical resolution methods. A library object, MatCarloRe, based on Simulink was created. Using the graphical interface of the Matlab simulator, it is possible to construct a DFT model of the process by using the library blocks. Each block carries the logic of a DFT gate. The Monte Carlo engine is based on the collection of the outputs of many runs and their agglomerate to construct significant statistics of interest.

The results of a case study based on the probabilistic assessment of the reliability of an alkylation and treatment olefin plant node were evaluated. Three commons problem of analytical methods (e.g. CTMC) are discussed: (a) the state explosion; (b) the impossibility of CTMC to evaluate the reliability of the system when basic events with fixed probability are considered; and (c) the impossibility of modularization of the tree in independent sub-trees without incurring in approximation due to a dynamic gate at the TE.

After presenting the validation of the MatCarloRe tool, based on simple cases, it is shown the resolution of the case study in three different configurations: (i) static configuration (FT); (ii) dynamic configuration (DFT); and (iii) dynamic configuration with the introduction of fixed probability basic events. As it is known the first two modeling case can be compared with analytical values calculated by use of combinatorial and CTMC methodologies, while the last case is solved only through the use of the simulation tool.

Section 2 presents an overview on the main limits of analytical methods; Section 3 make a comparison between traditional Monte Carlo reliability simulation and our approach; Section 4 introduce in more detail the MatCarloRe tool, its object library, and some validating example; in Section 5 is presented the results of the case study; and in Section 6 conclusions are reported.

2. Analytical reliability evaluation for DFT

A fault tree is a stochastic model for reliability evaluation based on the Boolean algebra. It can synthesize the ways of failures or the undesired events of a system. It is composed of entities known as "gates" which implement the Boolean relationships among the events, leading to the occurrence of "higher" events up to the tree. The inputs of the gates can be other gates or basic events (BE); the main undesired event is called the top event (TE) of the tree. The methodology is based on three assumptions: (i) the events are binary, (ii) the events are statistically independent, and (iii) the relationships between events are described with the Boolean logic through the gates (AND, OR and Voting).

The main constrains of traditional FTs is that dynamic time dependencies cannot be modeled and complex system behavior evaluated. DFTs were introduced to overtake these limitations: it is an evolution of traditional FT, in which dynamic gates are added. In particular if a FT includes at least one dynamic gate it becomes a dynamic-FT.

The most likely method used to solve a DFT is through the use of CTMC. They are indeed effective in representing various types of dependencies. The main problem of this methodology is the well known state space explosion which grows exponentially with the increasing of the number of Bes in the model. Moreover, the analysis of large DFT by the mean of CTMC can be costly in computational terms and in terms of modeling efforts.

Modularization is one way to tackle the analysis of large FT. For traditional FT, a module is simply a sub-tree which events do not occur in other parts of the tree. This technique intends to break down large models with a hierarchy of sub-models, characterized by independent failure modes. The sub-models are then analyzed in order to extract the measures of interest which are used as input as a simplified version of the sub-tree at the higher level of the hierarchy. The main point of this technique is the sintering of the equivalent failure rates to use in the simplified parent tree (Chatterjee, 1975; Khoda, Henley, & Inoue, 1989; Rosenthal, 1980). Dugan, Bavuso, and Boyd (1992), Dugan, Sullivan, and Coppit (2000), and Meshkat, Dugan, and Andrews (2002) have shown that it is possible to identify sub-trees and use several independent Markov chains for each of them. However, this approach has limitations when dynamic gates are present at high levels of the hierarchy. In fact, the modularization of dynamic sub-trees introduces errors due to the generalized probability distributions that, in the composed model, is still treated as negative exponential distribution (Anand & Somani, 1998; Huang & Chang, 2007). Other ordering methods aim to reduce the complexity of the CTMC by lumping and condensate chain states. However, this requires the construction of the entire chain and the subsequent ranking and aggregation, not solving the tedious problem of modeling the full chain (Feinberg & Chiu, 1987; Lanus, Yin, & Trivedi, 2003; Malhotra & Trivedi, 1993).

Another of the limitations resulting from the use of CTMC is the need to use a negative exponential distributions for each basic event considered. This is often simplistic in terms of modeling because events or components are often characterized by other distribution probabilities.

We therefore emphasize that many of the methods to solve the DFT are related to specific problems and can be difficult to generalize for all scenarios.

3. Reliability Monte Carlo simulation for DFT

Simulation programs, especially if well structured, are in general very comprehensible and known for the ease with which modifications and additions can be made. Perhaps their main benefit arises from the fact that it is possible to output information about sub-systems and gain more understanding of the whole system (Windebank, 1983). Monte Carlo simulation is a valuable method which is widely used in solving real problems in many engineering fields. It has been used by Goldfeld and Dubi (1987) and Dubi (1989) for the study of system reliability, based on the well-developed neutron transport ideas. The simulation technique allows the estimation of reliability indices by simulating the actual process and random behavior of the system through a computer model. This model has to be able the realistic scenario of the system lifetime.

Monte Carlo simulation is implemented by running the model a large number of times, each representing an ensemble of random

walks within the discrete phase space of system configurations, in order to generate a large number instances from which all the reliability indices required about the system are retrieved.

As the system is composed of many components (or elements) grouped together to perform a certain function, the system modeling begins with the identification of the components of which the system is composed. Let us denote by the possible states in which the ith component of the system may be. In the simplest case may have two possible states: one is the "up" or "active and functioning" state; the other is the "down" or "failed" state. The state of the system can be described by a vector S:

$$S = (s_1, s_2, \ldots, s_i, \ldots, s_n) \quad (1)$$

where n represents the number of components of the system. When a component changes its state, a new point in phase space is reached. Some of this point are of failure for the whole system. While in static-FT certain combination of component failures correspond to system failure, in DFTs the same state can be either a failure state or a good one, depending on the previous state sequence. Hence, in DFT it is not possible to define a system state a good one or a failed one without looking at the system evolution. Vector S changes with respect to time, so each simulation consist of a collection of state vectors representing the movement of the random walk within the phase space. Such history can be written as:

$$H_k = (S_0, t_0; S_1, t_1; \ldots; S_m, t_m) \quad (2)$$

where represents the simulation-run index t_i, the time of transition from state S_{i-1} to state S_i and represents the index of the ending simulation time. The point S_i, t_i represents a phase space point, while the collection of all such points is called the phase space of the system. This space is continuous in time and discrete in the states. The transitions among system states depend on the component transitions from the up state to the down one, according to a stochastic law (the probability density function) that characterizes the component behavior.

The only information needed for the analysis are: (a) the probability density function (pdf) of the time to failure of each component and their parameters values; (b) the mission time of the system; and (c) the system failure mode configuration.

The most common way to conduct the simulation of a FT is to consider the system as a whole; previous literature (Dubi, 1986; Lewis & Bohm, 1984) consider a system failure rate as the sum of all the component failure rate and calculate the transition time to the new state. After that, the component which performs the transition is chosen in a stochastic manner. When the component is chosen its failure rate is set equal to zero and the process is repeated till the occurrence of the TE or the end of the mission. The system operability (i.e. the occurrence of the TE) is evaluated each time a transition occurs. Generally it is convenient to make use of the FT methodology to check the system operability each time it changes its state. The problem with this approach is that we must check the operability state of the system each time the system makes a transition; moreover it is needed to recalculate the system failure rate by imposing the failed component failure rate equal to zero. When considering DFTs we need to take into account the system evolution history in order to assess the nature of the reached state. Moreover in the case of components with different time distribution the traditional indirect Monte Carlo method (Wu & Lewins, 1992), in which the transition time for the whole system is firstly sampled from the system transition time distribution and then the kind of transition is sampled, becomes impractical.

A more straightforward method to account for time-dependent failure and repair behaviors of components is the direct Monte Carlo simulation (Zio, 1995). It samples the time to failure of each BE instead of the overall system transition time.

Following the idea of direct Monte Carlo, our simulation approach makes use simultaneously of the FT and the Monte Carlo methodologies. Instead of simulating the system walk in the phase space we consider BEs as basic entities. The failure time of each BE is calculated and these information are passed to the gates which they are connected with. The gate state is determined and, if failed, its failure time is passed to the higher levels. In this way for each gate is possible to calculate if a failure occurred before the mission time and pass the failure time to next level. These information are important for the dynamic gates. Moreover many simulation data can be stored in a very straightforward way: for each gate we can obtain the failure number of occurrences, the mean time to failure and information about which connected subsystem forced mostly the failure of the gate.

The approach followed is very suitable for dynamic gates since the failing order is tracked. Therefore, in this way it is not necessary to store the previous system states and check the order of occurrence to asses whether a state is of failure or good.

The complexity of the algorithm is carried by each single gate, whose logic is programmed in order to infer its own state. For the same reason, with this approach there is no need to update the overall system failure rate at each transition.

Generated all the transition times of each component, the failure time for each gate can be calculated based on the logic of the gate itself with respect to its inputs. At the highest level information about the system state are available and used to compute the system reliability estimate.

Due to the nature of the approach, it would be straightforward to program or modify the logic of the gates in a modular way, without compromising the environment already set. In this way, the end user can just make full use of the gates created by the programmers and exploit with some small knowledge of Simulink the power of the library.

4. MatCarloRe library

The simulation tool implemented make use of a high level modeling interface: it allows the user to assemble the FT by picking basic events and gates from a library and dropping these elements on a graphical interface furnished by Simulink®. BEs and gates are then linked together to create the system configuration. The tool consists of a Simulink® library called MatCarloRe (Fig. 1), formed of blocks representing the various parts of DFT, such as the dynamic gates PAND, SPARE, SEQ and FDEP as well as the static gates AND, OR and Voting. In addition to the gates it is necessary to insert a block that calculates the failure times of BEs. Each block

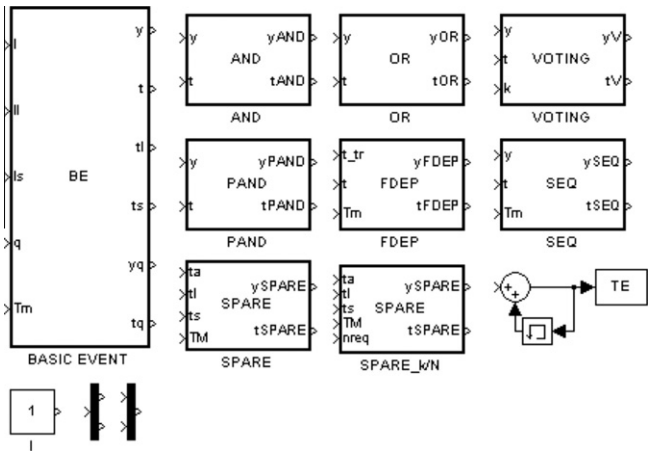

Fig. 1. MatCarloRe library elements.

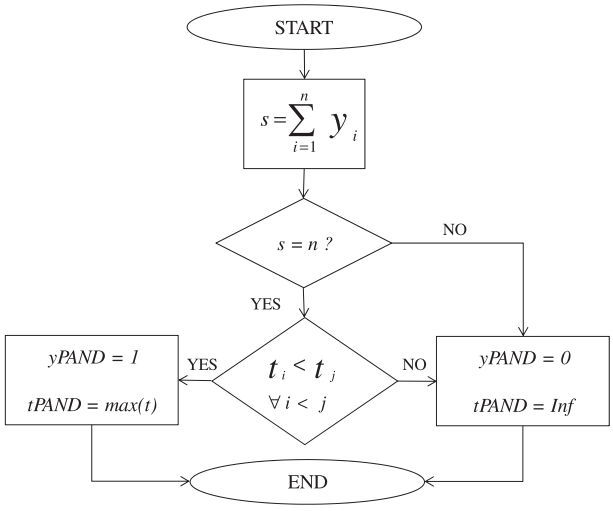

Fig. 2. Flow chart of the PAND block.

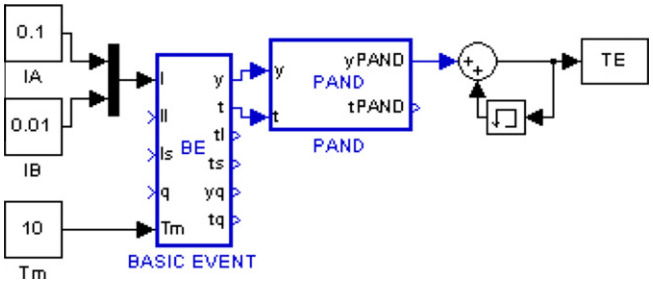

Fig. 3. Simulation model of a PAND gate with two basic event with failure rate $\lambda = 0.1, 0.01$ and mission time $t_m = 10$.

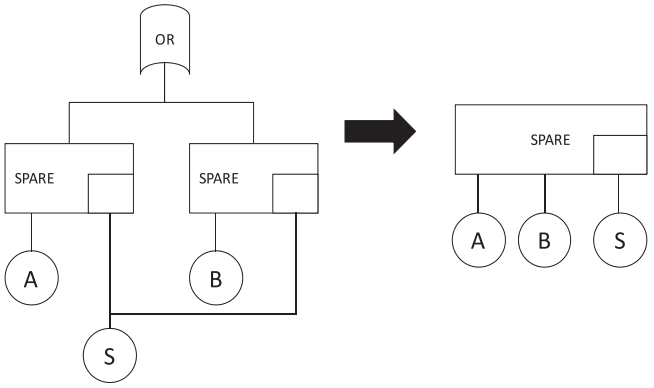

Fig. 5. Illustration of SPARE gates modeling vantages introduced by the tool; case of common shared spare. Left: relex model; right: MatCarloRe model.

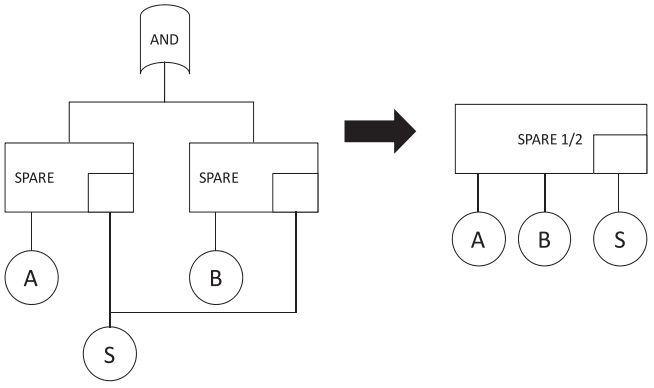

Fig. 6. Illustration of SPARE gates modeling vantages introduced by the tool; case of redundancy in active components. Left: relex model; right: MatCarloRe model.

representing a tree gate can receive *n* input and distribute *m* output by simply using the *mux* and *demux* blocks available in the main Simulink library. Inputs and outputs are of two kind: we define *y* as the binary vector which indicates whether the input and output have occurred (value 1) or not (value 0), and *t* as the vector containing the failure times. Only for the block representing the SPARE and the FDEP gates is not provided the input vector *y*.

Once the model is built and the input parameters are defined it is possible to run the simulation without a limit of iterations to achieve the desiderate estimation error. At each iteration the model returns a binary value that indicates whether the system has reached the state of failure or not. In particular, the model returns the value 1 if the fault is reached, 0 vice versa. To avoid large amounts of storage of the entire iterations binary vector the Simulink® block *memory* is used to set the progressive sum of results of

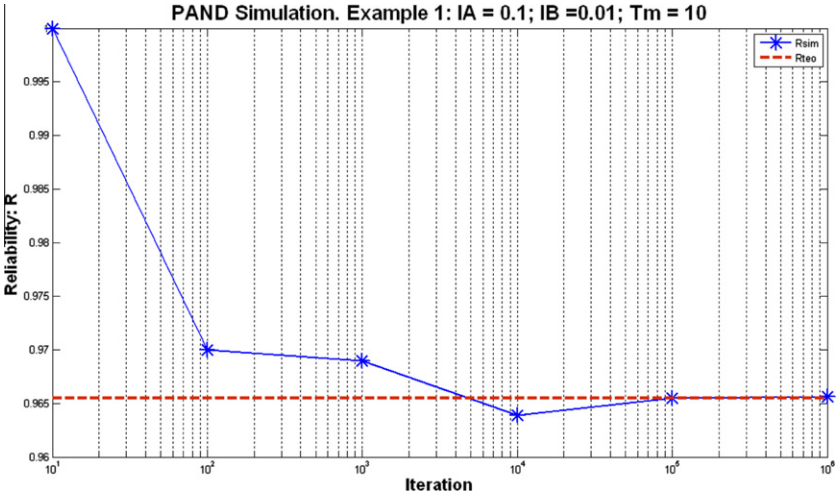

Fig. 4. Simulated vs. analytical reliability of a PAND gate with two basic event with failure rate $\lambda = 0.1, 0.01$ and mission time$t_m = 10$. Iteration are chosen equal to 10^i with $i = 1,2,\ldots,6$. Dotted line: analytical result; marked line: simulated results.

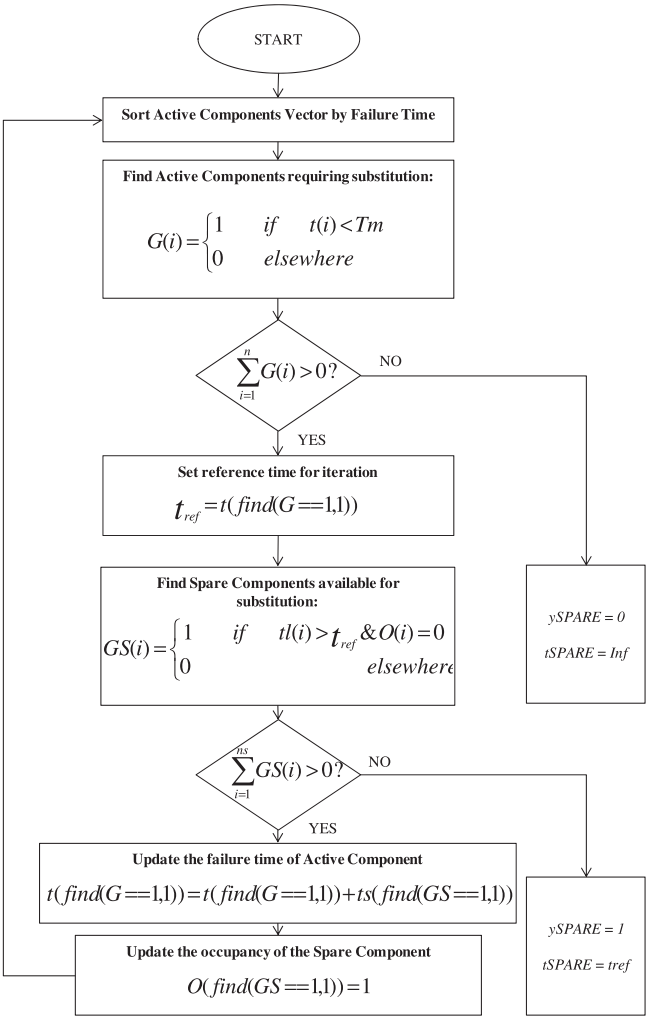

Fig. 7. Flow chart of the SPARE block.

each iteration. In this way only the total value of runs that revealed a fault is considered. The estimated unreliability of the system can be finally obtained dividing the number of runs which had shown the system failure by the total number of runs performed.

In the next section it is shown with small examples how to build a simulation model for a DFT introducing in more details the BE block and the dynamic gates. In order to proof the validity of the tool the results performed with the MatCarloRe are compared with analytical results.

4.1. BE block

The BE block is designed to generate the times of failure of basic events. It can generate the times of failure for components with negative exponential failure distribution (e.g. components subject to random failures) as well as fixed probability (e.g. the non-action of an operator). It is worth to mention the possibility to define events that follow distributions of any kind, such as Weibull, Normal distribution etc. simply by making small changes to the code in the BE block.

For components subjected to random failures the unreliability at time t can be expressed by the following relation:

$$F(t) = 1 - e^{-\lambda t} \tag{3}$$

where λ is the failure rate of the component. The simulated failure time can be calculated by the inverse relationship:

$$t^* = -\frac{log(1-F^*)}{\lambda} \tag{4}$$

where F^* is a random number generated in $[0,1]$. If t^* is smaller than the mission time, the component is considered as failed.

For events supplied with a fixed probability q the following procedure is defined: a random number q^* generated uniformly in $[0,1]$ is extracted and compare with the value q. This comparison returns a failure time according to the following relation:

$$t^* = \begin{cases} +Inf & if\ q^* \geqslant q \\ t_q^* & if\ q^* < q \end{cases} \tag{5}$$

where t_q^* is a random number generated uniformly in $[0, t_m]$.

4.2. PAND block

The PAND block models the logic that underlies the PAND gate of a DFT (Boudali et al., 2007; Dugan et al., 1992; Durga Rao et al., 2009). The logic of the gate can be summarized as follows: the occurrence of the gate is obtained if all input components have failed before the mission time but in a fixed order (i.e. from left to right in the graphical representation). The block logic is illustrated in the flowchart in Fig. 2. First, according to the vector y, it is verified if all the input events occurred. If this condition is not satisfied the gate does not trigger. Otherwise the following conditions are checked: $t_i < t_j$ for $\forall i < j$ with $i, j = 1,2,\ldots,n$, where is the gate number of inputs. If such control over failure times is satisfied the gate triggers with a time to failure equal to the maximum failure time of its inputs.

To test the validity of the block to perform the requested calculation we show the results of the following example. A PAND gate with two basic events is considered. The basic event failure rates are $\lambda = 0.1$ for the first component from the left and $\lambda = 0.01$ for the second one. The mission time is $t_m = 10$. In Fig. 3 is shown the model built for the simulation with the MatCarloRe tool. It is possible to see the BE block which takes as inputs the failure rates of the two components and the mission time of the system. The output of the block is the time to failure of the two components (i.e. vector t) and the binary vector y which indicates whether the time to failure of the components is smaller than the mission

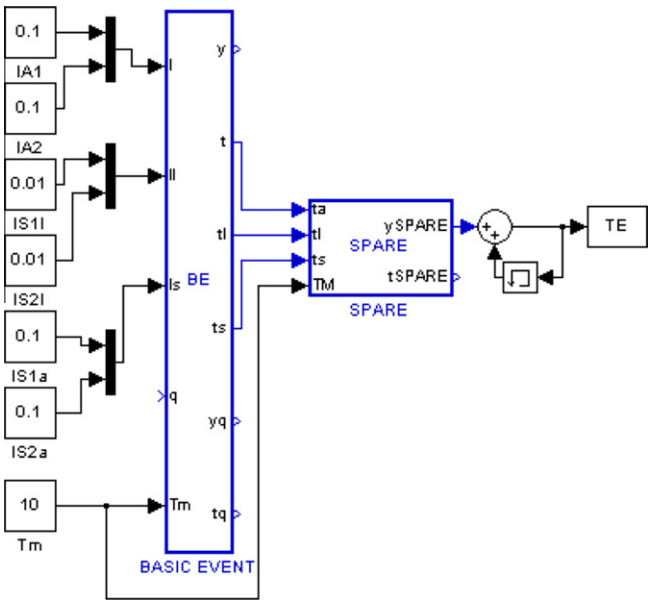

Fig. 8. Simulation model of a SPARE gate with two active components with failure rate $\lambda = 0.1$, two spare components with failure rates $\lambda = 0.1$, latency factor and mission time $t_m = 10$.

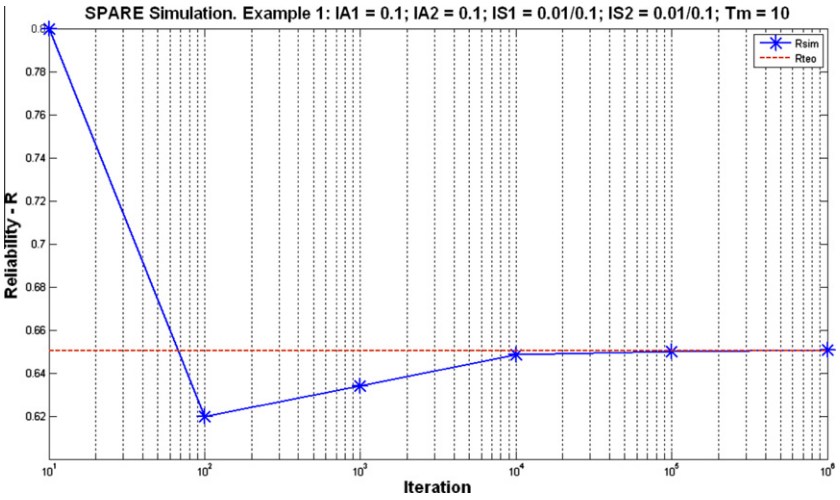

Fig. 9. Simulated vs. analytical reliability of a SPARE gate with two active components with failure rate $\lambda = 0.1$, two spare components with failure rates $\lambda = 0.1$, latency factor $a = 0.1$ and mission time $t_m = 10$. Iteration are chosen equal to 10^i with $i = 1,2,\ldots,6$. Dotted line: analytical result; marked line: simulated results.

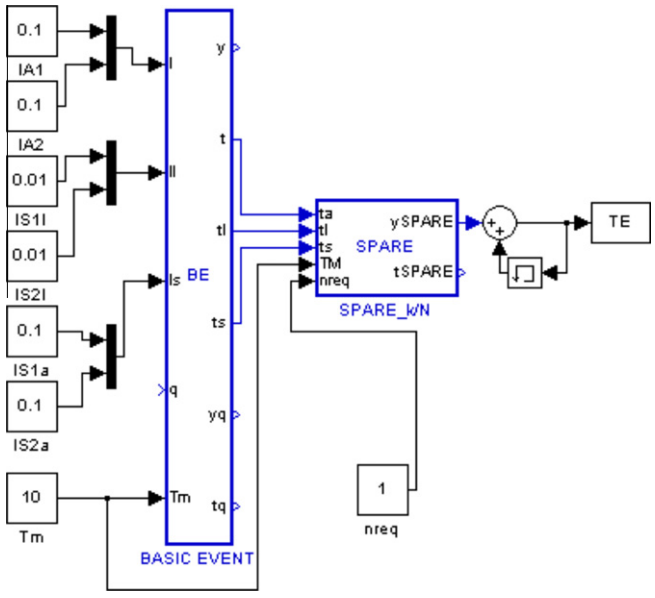

Fig. 10. Simulation model of a SPARE_k/N gate with two active components with failure rate $\lambda = 0.1$, two spare components with failure rates $\lambda = 0.1$, latency factor $a = 0.1$ and mission time $t_m = 10$; $nreq = 1$.

time. These vectors are given as input to the PAND block. The output scalar *yPAND* of the PAND block is passed to a progressive sum and at the end of all the iterations the result is stored by the block TE into the Matlab® Workspace.

The simulated reliability versus the analytical one calculated by the mean of the associated CTMC are shown in Fig. 4. Iteration are chosen equal to 10^i with $i = 1,2,\ldots,6$. It is evident that for a large number of iterations (of five magnitude order) the error is very low and acceptable.

4.3. SPARE block

The SPARE block models the logic that underlies the SPARE gate of a DFT (Boudali et al., 2007; Durga Rao et al., 2009). It can be summarized as follow: given n active components, if one of them fails it can be replaced by one of the ns spare parts. The spare parts can fail under two conditions: either because of a failure when in latent state or because of a failure after becoming active. The spare becomes active when it replaces a failed active component or a failed active spare. The failure of the gate occurs when the number of surviving components is less than the number of required components which depends on the logic of the gate (usually it is assumed equal to the initial active components number). The block can, therefore, model the logic of cold stand-by (i.e. spare parts cannot fail during the latent time) or warm/hot stand-by (i.e. when it is possible for a spare to fail even if not running; usually with a lower failure rate).

The block logic is illustrated in the flowchart in Fig. 5. Firstly it is needed to compute the time to failure of the components by the BE block. This is done for active components as well as for spare ones (both when active and in latent state). The SPARE block performs a permutation of the vector t, the time to failure of active components, sorting the vector in ascending order. Then, it is examined among active components whether there are components which have failed before the end of the mission time. If no failure is verified the gate will not trigger.

Vice versa, the algorithm checks if there are spare parts able to replace the active component that have failed. Let us consider the logic of replacement of a generic active component which fails. Its replacement can take place only if:

1. the spare part is still available (namely, it has not been used to replace another failed component) and;
2. the time to failure of the spare (during its latent condition) is greater than the time to failure of the active component to be replaced.

If these two conditions are not satisfied by any spare the gate triggers with a time of occurrence equal to the last failed active component. Otherwise the block updates the time to failure of the active component by adding the time to failure (when active) of the spare component chosen for the replacement. The substituting spare is finally declared as busy. The search for active components applicants for replacement is repeated till there are active components with failure time smaller than the mission time.

It is worth to highlight that the order in which the spares are chosen to replace the failed component follows the graphical order of positioning defined into the spare (e.g. in case of two spares that can replace an active failed component, it is chosen to replace the component with the spare that graphically is placed to the left).

This block offers many advantages in terms of modeling. Many reliability tools (Relex, Galileo Sullivan et al., 1999) are, in fact,

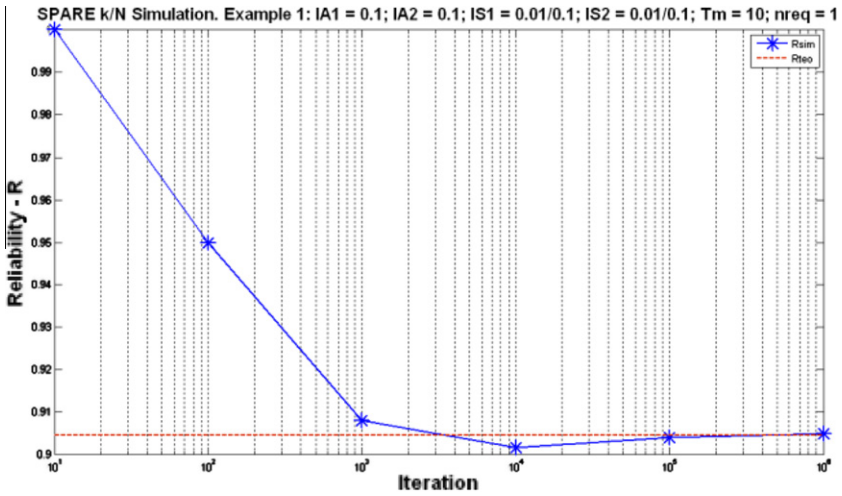

Fig. 11. Simulated vs. analytical reliability of a SPARE k/N gate with two active components with failure rate $\lambda = 0.1$, two spare components with failure rates $\lambda = 0.1$, latency factor $a = 0.1$ and mission time $t_m = 10$; $nreq = 1$. Iteration are chosen equal to 10^i with $i = 1,2,\ldots,6$. Dotted line: analytical result; marked line: simulated results.

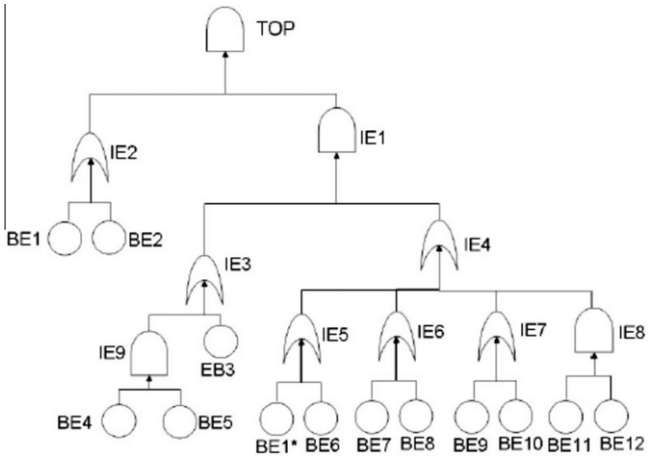

Fig. 12. FT of the section plant considered.

Table 1
Input data for basic events of the FT of Fig. 12.

ID	λ	q
BE1	–	$1.0 \cdot 10^{-5}$
BE2	$9.1 \cdot 10^{-4}$	–
BE3	–	$1.0 \cdot 10^{-7}$
BE4	$1.7 \cdot 10^{-4}$	–
BE5	$7.5 \cdot 10^{-4}$	–
BE6	$9.1 \cdot 10^{-4}$	–
BE7	$4.5 \cdot 10^{-3}$	–
BE8	$8.6 \cdot 10^{-3}$	–
BE9	$4.5 \cdot 10^{-4}$	–
BE10	$7.9 \cdot 10^{-3}$	–
BE11	$1.5 \cdot 10^{-4}$	–
BE12	$9.5 \cdot 10^{-4}$	–

difficult about the construction of DFT whit SPARE gates: if a spare part is shared among more components, the DFT will have as many SPARE gates as the number of active components which share the spare. In this way, the first input of the generic SPARE gate is the active component, while the second input is the shared spare part, common to all the set of SPARE gates drawn. The final logic implemented by the set of SPARE gates is realized linking them together by an OR gate in the upper level (Fig. 5).

Another situation is redundancy in the active components (i.e. not all the active components are requested for the system to work). In this case an AND gate is placed to the upper level (Fig. 6). If the number of active components is even greater than the presented examples in the previous figures the modeling activity is even more complex involving the use of AND and OR gates in the tree structure. Therefore what the SPARE gate of the MatCarloRe library is able to do is to bypass all these architectural tricks, by the simple use of a single block called SPARE_k/N, where k is the number of components requested to work and N is the number of initial active components (Fig. 6).

The differences in the flow chart between the two SPARE gates considered are located in the first rhombus of the chart in Fig. 7 where it is needed to consider the $N - k$ failures allowed for the system to work.

To test the validity of the two blocks SPARE and SPARE_k/N the results of two simple example are shown. The system is composed of two active and two spare components with failure rate $\lambda = 0.1$. The latency factor for the spare components is $a = 0.1$ and the mission time is $t_m = 10$. Figs. 8 and 10 show the models built with the MatCarloRe tool in the two different cases. The simulation models are equal in both cases except that the number of components needed for the system to work in the second example are defined by *nreq*.

The BE block takes as inputs the failure rates of the two active components, the failure rate of the spare components when in latent state and when active and the mission time of the system. It returns as output the time to failure of the two active components stored in the vector *t*. The time to failure of spare parts when in latent state and when active are given respectively in the vectors *tl* and *ts*. Vectors *t*, *tl* and *ts* and the mission time are the input of the SPARE block. The output *ySPARE*, taken over repeated iterations, is then used to compute reliability value.

In Fig. 9 is shown the simulated reliability versus the analytical calculated by the mean of the associated CTMC. Iteration are chosen equal to 10^i with $i = 1,2,\ldots,6$ It is evident that for iteration of order 10^5 the error of prediction is very low and acceptable.

The results of the simulation of the *k*/*N* model are shown in Fig. 11. Again for iteration of order the simulated reliability is very close to the CTMC results.

4.4. SEQ block

The feature of the SEQ gate is to force the components – inputs of the gate – to move towards the state of failure in a fixed order

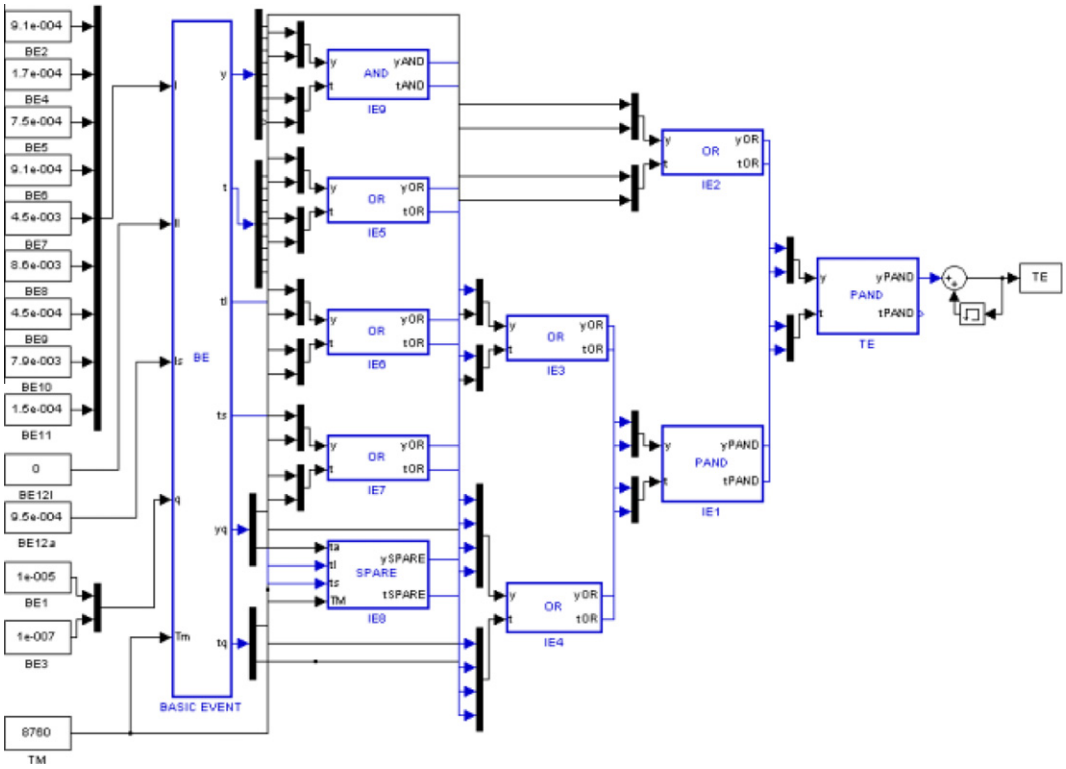

Fig. 13. MatCarloRe model of the DFT of Fig. 12. Case (iii).

Table 2
Unreliability and relative error for the model MatCarloRe of the FT in Fig. 12. Case (i): SFT with fixed probability for BE1 and BE3; Case (ii): DFT with failure rate for BE1 and BE3; Case (iii): DFT with fixed probability for BE1 and BE3.

Iter	Case (i)	Err. rel (%)	Case (ii)	Err. rel (%)	Case (iii)
10^1	0.7000	9.45	0	100.00	0
10^2	0.8000	3.48	0	100.00	0
10^3	0.7730	0.01	0	100.00	0
10^4	0.7717	0.18	1.00E−04	88.39	0
10^5	0.7734	0.04	6.00E−05	13.03	3.00E−05
10^6	0.7733	0.02	6.20E−05	16.80	4.40E−05
10^7	0.7732	0.02	5.58E−05	5.12	5.49E−05
10^8	0.7731	0.00	5.36E−05	0.98	5.45E−05
Fteo	0.7731		5.31E−05		

(Boudali et al., 2007; Dugan et al., 1992; Durga Rao et al., 2009). This order is usually expressed graphically by the position of the gate inputs, from left to right. It is generally used to represent different levels of degradation of a component. Therefore, a condition for the gate to trigger is the occurrence of all its inputs. The algorithm used for this task is simple: the SEQ block firstly calculates the sum S of the time to failure of all its inputs. If S is smaller than the mission time the gate triggers with a time to failure equal to S.

4.5. FDEP block

The FDEP block models the FDEP gate of a DFT (Boudali et al., 2007; Dugan et al., 1992; Durga Rao et al., 2009). The feature of this gate is to force the input components to reach the failure state if the trigger event has occurred before they fail by themselves. The block checks if the failure time of each input component is smaller than the trigger failure time. If the condition is true the component will fail with its own failure time. Vice versa the component will occur with failure time equal to the trigger failure time.

In the construction of a model with a FDEP, each component subjected to the action of the trigger is firstly connected to the FDEP gate and then the output of the latter is passed to the gate interested by the given component. We do not show the flow chart due to the simplicity of the task performed by the block. Likewise we do show any example for the FDEP block due its similarity with an OR gate between the trigger event and any of the basic event of the gate.

5. Study case

In this section we present a case of study of a real complex system, in order to demonstrate the effectiveness of the simulation tool to calculate the reliability of such systems. The case of study represents the FT model of a plant section for the alkylation and treatment of light olefin. Following the top-down procedure of the FT analysis, the tree was designed. The static-tree structure is shown in Fig. 12 and Table 1 reports the component failure rates.

Beside that model, the DFT was designed in order to consider a more realistic safety behavior that the plant exposes. The dynamic re-arrangement considered concerns the modeling of the gates IE1, IE8 and of the TE. In fact, in the static modeling they are represented with the traditional AND gates. That results in an approximate evaluation of the logic for the real system, since time dependencies cannot be considered with the static-FT. In the DFT, the gate IE8 was substituted with a SPARE gate as in a classic cold stand-by redundant configuration. The second re-arrangement is done by substituting IE1 with a PAND, in order to consider the priority condition that IE3 has on IE4. The same process is applied at the TE gate.

Therefore, three cases were studied: (i) simulation of the static-FT, (ii) simulation of the DFT without fixed probabilities by substituting with the relative failure rate calculated through (3) (i.e. assuming the value of F equal to q and calculating the failure rate

trough inverse relationship); (iii) the simulation of the DFT with the original parameters of Table 1.

The analytical resolution of these three cases expose different levels of complexity. In fact, the case (i) is the simplest because no time dependencies arise and traditional techniques, based on the Boolean algebra, can be used. Case (ii) introduces two kind of dynamic gates. One of them is placed at the TE. It makes impossible the use of techniques to relax the complexity of the model (e.g. modularization Chatterjee, 1975; Dugan et al., 1992; Dugan et al., 2000; Khoda et al., 1989; Meshkat et al., 2002; Rosenthal, 1980) without incurring in approximated calculation. The case (iii) can be classified as the most complex since it cannot be solved with the use of the traditional CTMC paradigm due to the presence of fixed probabilities. For this last case no analytical result are presented. The model of the case (iii) implemented in the MatCarloRe tool is shown in Fig. 13.

We conducted eight simulation for each case (see Table 2). The number of iterations were chosen till the maximum value of 10^8. For cases (i) and (ii) analytical results were computed through Relex®. The unreliability of the simulation model converges to the analytical result with 10^3 iterations in the case (i) with a very small relative error. In the case (ii) more iterations are needed to obtain valid results because of the more complex nature of the system involving temporal dependencies. About 10^8 iterations to achieve a small estimating error. In the case (iii) we claim that the number of iterations needed to achieve a small error in case (ii) could be used as well. This is supported by the fact that the unreliability seems to stabilize around 10^8 iterations.

6. Conclusion

In this paper we summarize an integrating technique of Monte Carlo simulation and FT methodology for reliability assessment of complex systems in presence of time dependencies. We showed that our simulating environment can go beyond the limitations of analytical methodologies with the additional advantage of a high level modeling interface based on the FT method. Some results are presented reporting good performance in terms of modeling and calculation efforts. Moreover important contributions for the SPARE model are introduced, reducing the efforts of the construction of a complex FT. It is shown that for iteration of order 10^7–10^8 it is possible to obtain reliable results for real system in a time frame of 1–10 h. Reducing computation time technique are well developed (e.g. biasing techniques) and could be easily introduced in the MatCarloRe tool. In the future our effort will be pushed in the development of tool characteristics for the calculation of important measures, availability and system performance indicators.

References

Amari, S., Dill, G., & Howald, E. (2003). A new approach to solve dynamic fault trees. *Annual Reliability and maintainability symposium*, 374–379.

Anand, A., & Somani, A. K. (1998). Hierarchical analysis of fault trees with dependencies, using decomposition. *Proceedings Annual on Reliability and Maintainability Symposium*, 69–75.

Bobbio, A., Portinale, L., Minichino, M., & Ciancamerla, E. (2001). Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. *Reliability Engineering and System Safety, 71*, 249–260.

Boudali, H., Crouzen, P., & Stoelinga, M. (2007). Dynamic fault tree analysis using input/output interactive Markov chains. In *Proceedings 37th annual IEEE/IFIP international conference on dependable systems and networks DSN '07, June 25–28* (pp. 708–717).

Cepin, M., & Mavko, B. (2002). A dynamic fault tree. *Reliability Engineering and System Safety, 75*, 83–91.

Chatterjee, P. (1975). Modularization of fault trees: A method to reduce the cost of analysis. *SIAM Reliability and Fault Tree Analysis*, 101–137.

Chiacchio, F., Compagno, L., D'Urso, D., Manno, G., & Trapani, N. (2011). Dynamic fault tree resolution: A conscious trade-off between analytical and simulative approaches. *Reliability Engineering and System Safety*. doi:10.1016/j.ress.2011.06.014.

Distefano, S., & Puliafito, A. (2007). Dynamic reliability block diagrams vs dynamic fault trees. In *Proceedings annual reliability and maintainability symposium RAMS'07, January 22–25* (pp. 71–76).

Dubi, A. (1986). *Monte Carlo calculations for nuclear reactors. CRC handbook of nuclear reactors calculations* (Vol. II). CRC Press.

Dubi, A. (1989). *Monte Carlo methods in reliability*. Operation Research and System Engineering Commission of the European Communities Joint Research Centre, Ispra Italy.

Dugan, J. B., Bavuso, S. J., & Boyd, M. A. (1992). Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability, 41*, 363–377.

Dugan, J.B., Venkataraman, B., & Gulati, R. (1997). Diftree: A software package for the analysis of dynamic fault tree models. In *Proceedings annual reliability and maintainability symposium, January 13–16* (pp. 64–70).

Dugan, J. B., Sullivan, K. J., & Coppit, D. (2000). Developing a low cost high-quality software tool for dynamic fault-tree analysis. *IEEE Transaction on Reliability, 49*, 49–59.

Dugan, J. B., Trivedi, K. S., Smotherman, M. K., & Geist, R. M. (1986). The hybrid automated reliability predictor. *Journal of Guidance, Control, and Dynamics, 9*, 319–331.

Durga Rao, K., Gopika, V., Sanyasi Rao, V. V. S., Kushwaha, H. S., Verma, A. K., & Srividya, A. (2009). Dynamic fault tree analysis using Monte Carlo simulation in probabilistic safety assessment. *Reliability Engineering and System Safety, 94*, 872–883.

Feinberg, B. N., & Chiu, S. S. (1987). A method to calculate steady-state distributions of large Markov chains by aggregating states. *Operations Research, 35*, 282–290.

Goldfeld, A., & Dubi, A. (1987). Monte Carlo methods in reliability engineering. *Quality and Reliability Engineering International, 3*, 83–91.

Huang, C. Y., & Chang, Y. R. (2007). An improved decomposition scheme for assessing the reliability of embedded systems by using dynamic fault trees. *Reliability Engineering System Safety, 92*, 1403–1412.

Khoda, T., Henley, E. J., & Inoue, K. (1989). Finding modules in fault trees. *IEEE Transactions on Reliability, 38*, 165–176.

Lanus, M., Yin, L., & Trivedi, K. S. (2003). Hierarchical composition and aggregation of state-based availability and performability models. *IEEE Transactions on Reliability, 52*, 44–52.

Lewis, E. E., & Bohm, F. (1984). Monte Carlo simulation of Markov unreliability models. *Nuclear Engineering and Design, 77*, 49–62.

Malhotra, M., & Trivedi, K. S. (1993). A methodology for formal specification of hierarchy in model solution. In *Proceedings fifth international workshop petri nets and performance models (PNPM-1993)* (pp. 258–267).

Marquez, A. C., Heguedas, A. S., & Iung, B. (2005). Monte Carlo-based assessment of system availability. A case study for cogeneration plants. *Reliability Engineering and System Safety, 88*, 273–289.

Marsaguerra, M., Zio, E., Devooght, J., & Labeau, P. E. (1998). A concept paper on dynamic reliability via Monte Carlo simulation. *Mathematics and Computers in simulation, 47*, 371–382.

Marseguerra, M., & Zio, E. (2004). Monte Carlo estimation of the differential importance measure: application to the protection system of a nuclear reactor. *Reliability Engineering and System Safety, 86*, 11–24.

Meshkat, L., Dugan, J. B., & Andrews, J. D. (2002). Dependability analysis of systems with on-demand and active failure modes, using dynamic fault trees. *IEEE Transactions on Reliability, 51*, 240–251.

Montani, S., Portinale, L., Bobbio, A., & Codetta-Raiteri, D. (2008). RADYBAN: A tool for reliability analysis of dynamic fault trees through conversion into dynamic bayesian networks. *Reliability Engineering and System Safety, 93*, 922–932.

Ren, Y., & Dugan, J. B. (1998). Design of reliable systems using static and dynamic fault trees. *IEEE Transactions on Reliability, 47*, 234–244.

Roberts, N. H., Vesely, W. E., Haasl, D. F., & Goldberg, F. F. (1981). *Fault tree handbook*, NUREG-0492. Washington: US NRC.

Rosenthal, A. (1980). Decomposition methods for fault tree analysis. *IEEE Transactions of Reliability, R-29*, 136–138.

Siu, N. (1994). Risk assessment for dynamic systems: An overview. *Reliability Engineering and System Safety, 43*, 43–73.

Sullivan, K. J., Dugan, J. B., & Coppit, D. (1999). The Galileo fault tree analysis tool. In *Proceedings of the twenty-ninth annual international symposium on fault-tolerant computing, June 15–18* (pp. 232–235).

Volovoi, V. (2004). Modeling of system reliability petri nets with aging tokens. *Reliability Engineering and System Safety, 84*, 149–161.

Windebank, E. (1983). A Monte Carlo simulation method versus a general analytical method for determining reliability measures of repairable systems. *Reliability Engineering, 5*, 73–81.

Wu, Y. F., & Lewins, J. D. (1992). Monte Calo studies of engineering system reliability. *Annual Nuclear Engineering, 19*, 825–859.

Zio, E. (1995). Biasing the transition probabilities in direct Monte Carlo. *Reliability Engineering and System Safety, 47*, 59–63.

Zio, E., Marella, M., & Podollini, L. (2007). A Monte Carlo simulation approach to the availability assessment of multi-state systems with operational dependencies. *Reliability Engineering and System Safety, 92*, 871–882.

Zio, E., Podofillini, L., & Levitin, G. (2004). Estimation of the importance measures of multi-state elements by Monte Carlo simulation. *Reliability Engineering and System Safety, 86*, 191–204.