RADYBAN: A tool for reliability analysis of dynamic fault trees through conversion into dynamic Bayesian networks

S. Montani, L. Portinale*, A. Bobbio, D. Codetta-Raiteri

Dipartimento di Informatica, Università del Piemonte Orientale, Via Bellini 25g, 15100 Alessandria, Italy

Available online 19 March 2007

Abstract

In this paper, we present RADYBAN (Reliability Analysis with DYnamic BAyesian Networks), a software tool which allows to analyze a dynamic fault tree relying on its conversion into a dynamic Bayesian network. The tool implements a modular algorithm for automatically translating a dynamic fault tree into the corresponding dynamic Bayesian network and exploits classical algorithms for the inference on dynamic Bayesian networks, in order to compute reliability measures. After having described the basic features of the tool, we show how it operates on a real world example and we compare the unreliability results it generates with those returned by other methodologies, in order to verify the correctness and the consistency of the results obtained.

1. Introduction

The modeling possibilities offered by fault trees (FT), one of the most popular techniques for dependability analysis of large, safety critical systems, can be extended by relying on Bayesian networks (BN) [1–5]. This formalism allows to relax some constraints which are typical of FTs. In addition, BNs allow to represent local dependencies and to perform both predictive and diagnostic reasoning.

In [6], we have shown how BNs can provide a unified framework in which also dynamic fault trees (DFT) [7], a rather recent extension to FTs able to treat several types of dependencies, can be represented.

In particular, DFTs introduce four basic (dynamic) gates: the warm spare (WSP), the sequence enforcing (SEQ), the functional dependency (FDEP) and the priority AND (PAND). A WSP dynamic gate models one primary component that can be substituted by one or more backups (spares), with the same functionality (see Fig. 1(a), where spares are identified by “circle-headed” arcs). The WSP gate fails if its primary fails and all of its spares have failed or are unavailable (a spare is unavailable if it is shared and being used by another spare gate). Spares can fail even while they are dormant, but the failure rate of an unpowered (i.e. dormant) spare is lower than the failure rate of the corresponding powered one. More precisely, being $\lambda$ the failure rate of a powered spare, the failure rate of the unpowered spare is $\lambda z$, with $0 \leq z \leq 1$ called the dormancy factor. Spares are more properly called “hot” if $z = 1$ and “cold” if $z = 0$.

A SEQ gate forces its inputs to fail in a particular order: when a SEQ is found in a DFT, it never happens that the failure sequence takes place in different orders. SEQ gates can be modeled as a special case of a cold spare [8], so they will not be considered any more in the following.1

In the FDEP gate (Fig. 1(b)), one trigger event $T$ (connected with a dashed arc in the figure) causes other dependent components to become unusable or inaccessible. In particular, when the trigger event occurs, the dependent components fail with $p_d = 1$; the separate failure of a dependent component, on the other hand, has no effect on the trigger event. FDEP has also a non-dependent output, that simply reflects the status of the trigger event and is called dummy output (i.e. not used in the analysis).

We have generalized the FDEP by defining a new gate, called probabilistic dependency (PDEP). In the PDEP, the

*Corresponding author. Tel.: +39 0131 360 184; fax: +39 0131 360 198.
E-mail addresses: stefania@mfn.unipmn.it (S. Montani), portinale@mfn.unipmn.it (L. Portinale), bobbio@mfn.unipmn.it (A. Bobbio), raiteri@mfn.unipmn.it (D. Codetta-Raiteri).

1The conceptual difference between the two kinds of gates is that the inputs to a SEQ do not need to be a component and its set of spares, but can be components covering any kind of function in the FT.
probability of failure of dependent components, given that
the trigger has failed, is $p_d \leq 1$.

Finally, the PAND gate reaches a failure state if and
only if all of its input components have failed in a
preassigned order (from left to right in graphical notation).
While the SEQ gate allows the events to occur only in a
preassigned order and states that a different failure
sequence can never take place, the PAND does not force
such a strong assumption: it simply detects the failure order
and fails just in one case (in Fig. 1(c) a failure occurs iff A
fails before B, but B may fail before A without producing a
failure in G).

The quantitative analysis of DFTs typically requires to
expand the model in its state space, and to solve the
the corresponding continuous time Markov chain (CTMC) [7].
Through a process known as modularization [9], it is
possible to identify the independent sub-trees with dynamic
gates, and to use a different Markov model (much smaller
than the model corresponding to the entire DFT) for each
one of them. Nevertheless, there still exists the problem of
state explosion.

In order to alleviate this limitation, as stated above, we
have proposed a translation of the DFT into a dynamic
Bayesian network (DBN). With respect to CTMC, the use
of a DBN allows one to take advantage of the factorization
in the temporal probability model. As a matter of fact, the
conditional independence assumptions implicit in a DBN
enable a compact representation of the probabilistic model,
allowing the system designer or analyst to avoid the
complexity of specifying and using a global-state model
like a standard Markov Chain; this is particularly
important when the dynamic module of the considered
DFT is significantly large.

In this paper, we describe RADYBAN (Reliability Analysis
with DYnamic BAyesian Networks), a tool we have
implemented able to realize an automatic translation of a
DFT into the corresponding DBN. The tool allows the
reliability engineer to access the modeling constructs of an
enhanced version of the DFT formalism for the construction
of the suitable reliability model; the resulting model is
then compiled in the corresponding DBN and the analysis
is performed in a transparent way to user, who has just to
specify the desired type of analysis algorithm.

The rest of the paper is organized as follows: In Section 2
we briefly review the basic framework of DBNs, in Section
3 we describe the main functionalities of RADYBAN, by
taking into consideration in particular the translation from
a DFT to a DBN for the computation of reliability
measures, and finally in Section 4, we show an application
of the tool features to a real world example taken from [2],
concerning an active heat reaction system. Conclusions and
future works are then reported in Section 5.

2. Dynamic Bayesian networks

DBNs [10] extend the BNs formalism by providing
an explicit discrete temporal dimension. They represent
a probability distribution over the possible histories of a
time-invariant process; the advantage with respect to
a classical probabilistic temporal model like Markov chains
is that a DBN is a stochastic transition model factored over
a number of random variables, over which a set of
conditional dependency assumptions is defined.

Time invariance ensures that the dependency model of
the variables is the same at any point in time. While a DBN
can in general represent semi-Markovian stochastic pro-
cesses of order $k - 1$, providing the modeling for $k$ time
slices, the term DBN is usually adopted when $k = 2$ (i.e.
only two time slices are considered in order to model
the system temporal evolution; for this reason such models are
also called 2-TBN or 2-time-slice temporal Bayesian
network).

Given a set of time-dependent state variables $X_1, \ldots, X_n$
and given a BN $N$ defined on such variables, a DBN is
effectively a replication of $N$ over two time slices $t$ and
$t + \Delta$ ($\Delta$ being the so-called discretization step usually
assumed to be 1), with the addition of a set of arcs
representing the transition model. Letting $X_i'$ denote
the copy of variable $X_i$ at time slice $t$, the transition model is
defined through a distribution $P[X''_{i,t+\Delta}|X'_{i,t}, Y', Y^{t+\Delta}]$ where
$Y'$ is any set of variables at slice $t$ other than $X_i$ (possibly
the empty set), while $Y^{t+\Delta}$ is any set of variables at slice
$t + \Delta$ other than $X_i$ ($Y^{t+\Delta}$ is non-empty only in the case of
the PDEP gate conversion). Arcs interconnecting nodes at
different slices are called inter-slice edges, while arcs
interconnecting nodes at the same slice are called intra-
slice edges. For each internal node, the conditional probabilities are stored in the conditional probability table (CPT) in the form \( P[X_{t+1}^t | X_{t}^t, Y_{t}^t] \).

Of course, a DBN defined as above (i.e. a 2-TBN) represents a discrete Markovian model. The two slices of a DBN are often called the anterior and the ulterior layer. Finally, it is useful to define the set of canonical variables as \( \{ Y : Y^{t} \in \bigcup_i \text{Parents}[X_{t+1}^t] \} \); they are the variables having a direct edge from the anterior level to another variable in the ulterior level. A DBN is in canonical form if only the canonical variables are represented at slice \( t \) (i.e. the anterior layer contains only variables having influence on the same variable or on another variable at the ulterior level). In the following we will consider DBNs in canonical form.

Given a DBN in canonical form, inter-slice edges connecting a variable in the anterior layer to the same variable in the ulterior layer are called temporal arcs; in other words, a temporal arcs connects variable \( X_{t}^t \) to variable \( X_{t+1}^t \). The role of temporal arcs is just in defining the nodes they are connecting as copies of the same variable at different slices. It follows that no variable in the ulterior layer may have more than one entering temporal arc.

**RADYBAN** explicitly uses the notion of temporal arc in its representation.\(^2\)

Concerning the analysis of a DBN, different kinds of inference algorithms are available. In particular, let \( X^t \) be a set of variables at time \( t \) and \( Y_{ab}^{t} \) any stream of observation from time point \( a \) to time point \( b \) (i.e. a set of instantiated variables \( Y_{i}^{t} \) with \( a \leq j \leq b \)). The following tasks can be performed over a DBN:

- **Filtering or Monitoring**: Computing \( P(X_{t}^{t} | Y_{1}^{t}) \), i.e. tracking the probability of the system state taking into account the stream of observations received.
- **Prediction**: Computing \( P(X_{t+h}^{t} | Y_{1}^{t}) \) for some horizon \( h > 0 \), i.e. predicting a future state taking into consideration the observation up to now (filtering is a special case of prediction with \( h = 0 \)).
- **Smoothing**: Computing \( P(X_{t}^{t-l} | Y_{1}^{t}) \) for some \( l < t \), i.e. estimating what happened \( l \) steps in the past given all the evidence (observations) up to now.

Different algorithms, either exact (i.e. computing the exact probability value that is required by the task) or approximate can be exploited in order to implement the above tasks.

Among those algorithms, particularly popular are the junction tree (JT) inference (based on the construction of a classical BN inference data structure called junction or join tree \([10,12]\) and belonging to the category of exact inference algorithms) and the Boyen–Koller (BK) algorithm \([13]\), a parameterized procedure that, depending on the parameters provided (disjoint sets of variables called clusters in the algorithm), may return exact as well as approximate results.\(^3\)

In the RADYBAN tool, the user can select either the filtering/prediction or the smoothing task, and for each given task she/he may decide to use either the JT or the BK implementation.

### 3. The RADYBAN tool

#### 3.1. Tool functionalities

The main features of RADYBAN allow the user to: (1) edit a (D)BN and draw inferences on it; (2) edit a DFT; and (3) automatically convert a DFT into the corresponding DBN, on which both predictive and diagnostic inference can then be drawn. Items (1) and (2) are described below; item (3), which represents the core contribution of this paper, is extensively treated in Sections 3.3 and 3.4. The tool architecture is depicted in Fig. 2.

#### 3.1.1. Editing a DBN

BNs (and DBNs in particular) can be directly used as a reliability modeling and analysis tool \([5,6,15–18]\). In this case, the reliability engineer can design the model by resorting to the basic formalism features. In particular, for DBNs:

1. for each system component, a variable is introduced at the anterior level and replicated at the ulterior level via a temporal arc connection;

\(^2\)A commercial tool using the same notion in modeling DBNs is BayesiaLab [11].

\(^3\)Very popular approximate algorithms are also those based on stochastic simulation [14]; one of such algorithms, called likelihood weighting, is used for instance in the BayesiaLab tool.
2. intra-slice dependencies are introduced and quantified;
3. inter-slice dependencies are finally modeled by connecting variables at the anterior layer with those variables at the ulterior layer depending on them (i.e. that are influenced by them) and by properly quantifying such a dependence.

For example, if the modeling follows a combinatorial style, like in FTs, the second point may possibly result in the addition of new variables representing particular functionalities of the modeled system (like standard gates in a FT) [1,5,16]. Both points 2 and 3 may then be involved if one has to model dynamic aspects of system functionalities like those represented by the dynamic gates of a DFT (see [7,8] and Section 3.3 for more details).

By means of the tool’s GUI, which has been built relying on the DrawNet graphical tool [19], the user is allowed to directly draw the desired DBN structure, by labeling nodes (variables) with the corresponding layer (0 for the anterior and 1 for the ulterior) and by using the notion of temporal arcs (depicted as thick edges in the figures) to identify temporal related copies of the same variable.

The CPTs can be easily inserted relying on a user friendly functionality, in which every row is automatically completed by calculating the last entry as the difference between 1 and the sum of the other values. Additionally, it is possible to identify query nodes (not necessarily the node corresponding to the top event (TE)—i.e. the global system failure, as usually required in (D)FT analysis), and to provide a stream of observations (i.e. evidences), each one labeled with its observation time.

When the network has been fully characterized, the user can choose to perform a filtering and prediction inference, or a smoothing inference, by setting the desired time horizon (i.e. the time point \(T\) up to which to compute results) and the desired time step \(k\).

The posterior joint probability of the set of query variables given the evidence is then computed starting from time 0 up to time \(T\), every \(k\) time instants. Notice that \(k\) (the time step used for the presentation of the results) is conceptually different from \(\Delta\) (the discretization step used to define the difference in time between the anterior and the ulterior layer of the DBN). This means that, if the system components have an exponential failure rate given in \(\text{fault/hour}\) and \(k = 10^3\) and \(\Delta = 1\), then if the anterior layer represents the system model at time \(t\), the ulterior layer models the system at time \(t + 1\), while the results will presented as: results at time \(t = 0\); results at time \(t = 1000\); results at time \(t = 2000\), etc.

Of course, the difference between a filtering and a smoothing inference relies on the fact that in the former case, while computing the probability at time \(t\) (\(0\leq t \leq T\)), only the evidence gathered up to time \(t\) is considered; on the contrary, in the case of smoothing the whole evidence stream is always considered in the posterior probability computation. It should also be clear that the specific task of prediction can be obtained by asking for a time horizon \(T\) greater than the last time point considered for an observation.

The classical computation of the unreliability of the TE is a special case of filtering, with an empty stream of observations.

Smoothing may be for instance exploited in order to obtain diagnostic knowledge (e.g. given that the system has been observed failed at time \(t\), to compute the probability of failure of basic components prior to \(t\)).

In the RADDYBAN tool, algorithms for filtering and smoothing on DBN [10] have been implemented by resorting to Intel PNL (probabilistic networks library), a set of open-source C++ libraries (see [20]), to which we have provided some minor adjustments.

3.1.2. Editing a DFT

Modeling the failure mode of a system as a DBN might be complicated for the user, while drawing the DFT model and generating automatically the corresponding DBN, is sometimes more practical. In this way, the DFT becomes a high level formalism allowing the user to express in a straightforward way the relations between the components of the system, whose modeling in terms of DBN primitives would be less comfortable.

The DFT editor allows the modeler to resort to standard DFT constructs (i.e. boolean and dynamic gates), as well as to specify additional properties for the analysis. In fact, the user may indicate which events will be queried and which events have been observed (true or false) at a given time point.

On the DFT, the user can also specify the analysis time step \(k\), as well as the mission time \(T\) and the inference algorithm to be adopted on the corresponding DBN for the required analysis (i.e. the analysis must be performed from time 0 to time \(T\) every \(k\) instants). This information is directly inherited by the corresponding DBN when translation is required. In this way, the DFT is exploited as an easy and well-known formalism, to which the user is typically already familiar, through which all the needed data for DBN inference can be given in input.

Particularly important from the quantitative analysis point of view is another parameter that the user can set on the DFT: the discretization step \(\Delta\). Since DBN is a discrete time formalism, a suitable discretization step must be defined in case failure specification on the system components are given in a continuous way. Let us suppose that a basic component \(C\) is characterized by an exponential failure rate \(\lambda_C\): given a discretization step \(\Delta\), we can characterize the failure probability of \(C\) as

\[
P[C\text{ failed at time } t|C\text{ working at time } (t - \Delta)] = 1 - e^{-\lambda_C \Delta}.
\]

In terms of the corresponding DBN, \(\Delta\) represents the amount of time separating the anterior layer from the ulterior layer.
There is a trade-off between the approximation provided by discretization and the computational effort needed for the analysis: smaller is the discretization step, more accurate are the results obtained (and closer to the continuous case computation), but greater is the time horizon required for the analysis (and thus the computation time). In fact, if failure rates are given as \( \text{fault/hour} \) and we set a mission time of \( T \) hours, a discretization step \( D = 1 \text{ h} \) will require analysis up to step \( t = T \), while a discretization step \( D = 10 \text{ h} \) will only require analysis up to step \( t = T/10 \) (since each step will count as 10 time units); this fact, in DBN inference, will result in a speed up of the result computation, because a smaller number of time slices have to be considered (i.e. a time slice in the latter case approximate 10 slices in the former).

3.2. Graphical interface description

Fig. 3 shows a screenshot of the graphical interface (DrawNet) of our tool. It is mainly composed by three windows; the Main window allows the user to draw the DFT model, while in the window named Property Page, it is possible to set the attributes of the node currently selected in the main window. From the Execute menu of the Main window, the user can run the conversion and the analysis of the DFT model. At the end of such process, the obtained results are displayed in the window called Solver Execution.

3.3. Translating dynamic gates

In this section, we present the conversion of the WSP gate in the corresponding DBN. The rules for converting the other gate types and the Boolean gates can be found in [1,6].

In a DFT, different configurations of WSP can be designed. Of particular interest are those in which the same pool of spares is shared across a set of WSP. In this case, each primary component is allowed to request the items in the pool in a precise order—if more than one is still dormant. As an example, let us consider a situation where two components \( A \) and \( B \) can be substituted by two spares \( S_A \) and \( S_B \) (see Fig. 4). In particular, \( S_A \) is \( A \)'s spare, and

![Image](image-url)
will substitute B only if: (i) A is working, and (ii) SB is failed. If A fails, it will request the activation of SA, and only if it is unavailable it will request SB. SB is B’s spare: analogous considerations hold. Every gate fails if and only if its primary component and all the available (i.e. working and dormant) spares in the pool fail. The structure of the DBN corresponding to this situation is shown in Fig. 5.

It can be observed that each component node at time \( t + \Delta \) depends on its copy at time \( t \) (temporal arcs). Moreover, each spare component also depends on the two primary components and on the other spare. Each spare is modeled as a random variable assuming four values, namely: (1) dormant, (2) operational on A, (3) operational on B and (4) failed. On the other hand, every primary component is a classical binary variable assuming two values: (0) working and (1) failed. If both primary components are working, each spare maintains a failure rate equal to \( z \lambda \); on the other hand, if A is down, SA switches to a failure rate equal to \( \lambda \) (since the spare is now in the active mode); the same happens if A is working, but B and SB are both failed. SB works similarly on its primary component B. As an example, Table 1 reports some entries of the CPT of SA, in the hypothesis that SA was dormant at time \( t \) and SB was failed. Note that the network complexity and the CPT dimensions would significantly increase as the number of shared spares increases, because the number of dependencies induced by the pool of spares increases as well. Each WSP gate (in the example, the one having A as its primary component, i.e. WSPA) and the one having B as its primary component, i.e. WSPB) is modeled as a deterministic AND node among its three inputs: the primary component and the two spares in the pool.

### 3.4. Combining the modules into a single DBN

Dynamic gates can be connected, in order to build a complex DFT. To understand what combinations can be modeled, and how we can provide an automatic translation of the DFT into the corresponding DBN, we have to recall how the gates themselves are meant to be applied.

In particular, according to [8]:

1. the dependent events of a PDEP (or FDEP) can only be basic events, which could be the input of another dynamic gate;
2. WSP can have only basic events as inputs and two or more spare gates can share some spare components; a set of WSP sharing a pool of spares is treated as a single module for translation (see Section 3.3);
3. PAND can have any type of input including outputs from other gates; two or more PAND can also be combined in a cascade manner.

These simplifications let us derive a general algorithm for building the DBN corresponding to the overall DFT. The procedure follows a modular approach, in the sense that it builds the output DBN by combining the various DBN fragments corresponding to the different gates.

From the structural point of view, combining different fragments is trivial; just overlap nodes corresponding to the same variable in different fragments. Fig. 6(a) shows an example of structural combination of the subnets of a WSP gate with one primary component \( P \), one spare B and a PDEP with a component T triggering B. The WSP gate is modeled as a deterministic AND of its input component B and P (see also [1,6]). The PDEP DBN (shown in the upper part of Fig. 6(a)), on the other hand, highlights an intra-slice dependency of B on the triggering event T, which in fact produces an immediate failure of its dependent component (with probability \( p_D \)). The PDEP gate simply mirrors the trigger status, and is not reported in the net (see [1,6] for details).

While the structural combination is relatively simple, the quantitative combination of the conditional probabilities (i.e. the generation of the CPTs relative to the combined structure) may be rather problematic. The problem stands in the fact that the structural combination will introduce new dependencies when overlapping nodes; the question is whether there exists a method of quantifying such dependencies in a modular way, by combining the CPTs of the original fragments, under a set of reasonable assumptions. This is a well-studied issue in BN theory under the name of “causal” or “conditional independence” [21,22]. The main point refers to the possibility of avoiding

---

Even if conceptually the spare component B should assume three values namely dormant, operational on P and failed, since B is not shared with other WSP gates, two values are sufficient (see [6,17] for details).
a complete CPT specification for a given node, when the number of the parents is too large for a reasonable assessment. Common for these approaches is the realization that all parameters are required if we do not make additional assumptions. However, if the domain experts are able to identify, e.g. functional relations, then this should be taken into consideration.

This can be explained by considering a structural transformation called divorcing [12]; it essentially consists in factorizing the assessment of the CPT of a given node with a large number of parents, by adding new parent nodes representing a set of the original parents and by considering their combination as a “noisy” (probabilistic) functional relation. An example is shown in Fig. 6(b), where parent nodes of node $B(t + A)$ in Fig. 6(a) are “divorced” by creating new parents $B'(t + A)$ and $B''(t + A)$. Conceptually, node $B'(t + A)$ represents the combination of nodes $B(t)$ and $T(t + A)$, while node $B''(t + A)$ is the combination of $B(t)$ and $P(t)$. A way of implementing this consists in setting four values for nodes $B'(t + A)$ and $B''(t + A)$ such that for example $B'(t + A) = \text{“00”}$ iff $B(t) = 0 \land T(t + A) = 0$, $B''(t + A) = \text{“01”}$ iff $B(t) = 0 \land T(t + A) = 1$ and so on. Node $B_t(t + A)$ implements the noisy relation used for integrating the original CPTs and determined by the underlying assumptions we want to make.

A typical example of such assumptions is the classical “noisy-OR”. Noisy-OR implies the independence of the causes that inhibit the presence of a given consequence and has a cumulative effect over the consequence (the probability of having the consequence when more than one cause is present is higher than the consequence’s probability when each cause is singularly present) [12,14].

For instance, let us suppose that in the example of Fig. 6(a), we quantify $P[B = 1 | T = 1] = p_d = 0.8$, $z = 0.5$, $\lambda = 0.1$, $e^{-z \lambda} \approx 1 - z \lambda$ and $e^{-z \lambda} \approx 1 - \lambda$, in the hypothesis that the failure rate is sufficiently small [5] and $A = 1$ (as usual $0 = \text{working}$ and $1 = \text{failed}$). With a noisy-OR interaction we could compute for example:

$$P[B(t + A) = 1|B(t) = 0, T(t + A) = 1, P(t) = 1] = P[B_t(t + A) = 1|B'(t + A) = \text{“01”}, B''(t + A) = \text{“01”}] = 1 - ((1 - p_d)(1 - \lambda)) = 1 - 0.2 \cdot 0.9 = 0.82. \quad (1)$$

Another functional relation that may be of interest in the reliability context corresponds to the situation where, given a set of potential causes of an effect, the most severe cause prevails over the others; this can be modeled by a...
functional node where the probability of a value is set to the maximum of the probabilities for such a value on the divorced parent nodes. Let us call this assumption MSP (most severe prevailing) interaction.

If we assume an MSP interaction, the CPT for node $B(t + \Delta)$ is obtained as in Table 4 (where only the probabilities of failure for $B(t + \Delta)$ are reported), by taking the maximum of the corresponding entries in Tables 2 and 3.

Table 2
Probability of failure of $B(t + \Delta)$ in the PDEP DBN

<table>
<thead>
<tr>
<th>$B(t)$</th>
<th>$T(t + \Delta)$</th>
<th>Failure of $B(t + \Delta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3
Probability of failure of $B(t + \Delta)$ in the WSP DBN

<table>
<thead>
<tr>
<th>$B(t)$</th>
<th>$P(t)$</th>
<th>Failure of $B(t + \Delta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4
Probability of failure of $B(t + \Delta)$ in the combined network

<table>
<thead>
<tr>
<th>$B(t)$</th>
<th>$T(t + \Delta)$</th>
<th>$P(t)$</th>
<th>Failure of $B(t + \Delta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$0.05 = \max(0.05, 0.05)$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$0.1 = \max(0.05, 0.1)$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$0.8 = \max(0.8, 0.05)$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$0.8 = \max(0.8, 0.1)$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\max(1, 1)$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\max(1, 1)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$\max(1, 1)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$\max(1, 1)$</td>
</tr>
</tbody>
</table>

This implies, for instance, that when the spare is operational, because of the failure of component $P$, if the trigger occurs, then it prevails over the usual degradation of $P$ (governed by failure rate $\lambda$), since the trigger failure probability is larger than the ones induced by the $P$’s failure rate.

Notice that the fourth row of Table 4 corresponds to the computation of Eq. (1); in the case of Eq. (1), the computed probability is slightly larger since in the noisy-OR interaction, the usual degradation of the spare is assumed to accumulate with the trigger influence, while in the MSP interaction the trigger influence completely hides spare’s degradation. This may introduce some approximation in the analysis results.

Finally, it is worth remarking that, in order to compute the entries of Table 4 there is no need to make explicit the divorcing structure of Fig. 6(b); once the modeler has decided the suitable noisy functional relation for components shared across different gates, this can be directly applied to the structure of Fig. 6(a). In the current example, our tool will in fact directly produce the structure of Fig. 6(a); at the current stage both the MSP interaction and the noisy-OR are implemented in our tool.

4. An example

The example we report is inspired from [2] and represents an active heat rejection system (AHRS). The block scheme of the AHRS’s architecture is depicted in Fig. 7; such system is composed by two redundant thermal rejection units $A$ and $B$, each one possessing a primary component ($A_1$ and $B_1$, respectively) and a cold spare ($A_2$ and $B_2$, respectively). $A_1$ and $B_2$ are powered by a common source $P_1$, which acts as a trigger in a FDEP gate, in which $A_1$ and $B_2$ are the dependent components. Similarly, $B_1$ and $A_2$ are powered by $P_2$. An extra stand-by cold spare unit ($S$) is shared between the two thermal rejection units $A$ and $B$, and is powered (and potentially triggered) by the source $P_3$.

The time to fail of any component in the system is a random variable ruled by the negative exponential

![Fig. 7. The block scheme of the AHRS’s architecture.](image-url)
distribution; Table 5 shows the failure rate of every component.

Fig. 8 shows the DFT, in canonical form, for the AHRS system. The DBN corresponding to the DFT in Fig. 8, is shown in Fig. 9 and is automatically generated given the DFT model, by using our tool. The nodes inside the DBN derive from the translation of the basic events and of the gates (both static and dynamic) inside the DFT; the translation of the internal events is not necessary [1].

After the conversion of the DFT in a DBN, we can perform the analysis of the latter by means of our tool. Table 6 shows the unreliability of the system versus the mission time varying between 0 and 100 h, with different discretization steps $(\Delta t = 1, 0.5, 0.05 \text{h}, \text{respectively},$ in columns 2, 3 and 4). To perform such a computation, we just used a filtering task by querying node $TE$ without providing any observation stream; in other words we performed standard prediction. The obtained results have been successfully verified by comparison with the results returned by other tools on the same DFT model. Such tools are DRPFTproc [23] (based on modularization [9] and conversion to Stochastic Petri Nets of dynamic gates) and Galileo [24] (based on modularization, binary decision diagrams (BDD) and CTMCs). Last two columns of Table 6 reports also the results obtained using such tools. It is easy to verify that, as the discretization step is reduced, RADYBAN results become closer and closer to the results obtained by means of the other tools, thus confirming the claim that, in this example, the only source of approximation in using DBNs is due to discretization. Moreover, as already mentioned, a trade-off between computation time and result precision exists: if approximated results are sufficient, a quicker DBN inference can be obtained by choosing a relatively large discretization step.

Note that in our example, there is the presence of FDEP gates; they are equivalent to PDEP gates with $p_d = 1$ (see Section 1). So in this case, the use of the MSP (see Section 3.4) does not determine any approximation in the analysis of the model: the use of the MSP when dealing with a FDEP gate is equivalent to deal with a noisy-OR gate.

DBN also offer additional analysis capabilities with respect to Markov models and Petri Nets: smoothing inference allows to rebuild the past history of the system, given a stream of observations. As an example, we have considered a situation in which the overall system was observed as operational at time $t = 10$ and 20 h, while it was observed to be failed ($TE = true$) at $t = 60$ h. By applying a smoothing algorithm, RADYBAN was able to provide the probabilities of failure of the system in the time span $20 \text{h} \leq t \leq 60$ h (see Table 7). For example, we can state that, by knowing that the system was certainly operational at $t = 20$ h and was certainly failed at $t = 60$ h, the probability that it was already failed at $t = 50$ h is about 0.4%. This suggest that a very unlikely event has occurred, because just 10 h before the observation of the failure, the system was almost definitely operational.

### Table 5

<table>
<thead>
<tr>
<th>Component</th>
<th>Failure rate ($\lambda$) (h$^{-1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>0.001</td>
</tr>
<tr>
<td>$A_2$</td>
<td>0.005</td>
</tr>
<tr>
<td>$B_1$</td>
<td>0.002</td>
</tr>
<tr>
<td>$B_2$</td>
<td>0.0035</td>
</tr>
<tr>
<td>$S$</td>
<td>0.005</td>
</tr>
<tr>
<td>$P_1, P_2, P_3$</td>
<td>0.003</td>
</tr>
</tbody>
</table>

Fig. 8. The DFT model of AHRS.
In the examples reported in this section exact algorithms (based on the calculation of the junction tree) were adopted, both for monitoring and for smoothing procedures.

5. Conclusions

In this paper, we have described RADYBAN, a tool that allows the user to draw a DBN and to ask for diagnostic or predictive inference on it, as well as to draw a DFT, obtain an automatic conversion into the corresponding DBN, and ask for reliability measures by means of DBN inference algorithms. We have described the tool’s functionalities, as well as the methodology underlying the translation of the input DFT into the DBN used for the analysis of the model.

To test both the proposed conversion methodology and the tool performance, we have run some examples, one of which has been presented in this paper: the results obtained using the DBN are basically identical to the ones obtained using other analysis techniques described in the literature. Our experimental results therefore demonstrate how DBN can be safely resorted to if a quantitative analysis of the system is required.

In the future, we plan to extend the tool capabilities, by adding ad hoc structures to the DFT, which can then be
naturally characterized in the corresponding DBN: for example, we will allow the insertion of multi-valued nodes, the modeling of repair policies and the specification of conditional dependencies among basic events.

A modularization procedure on the DFT (similar to that proposed in the Galileo tool) is currently under examination, in order to improve the inference computational time on the DBN in case of independent parts (modules) of the input DFT; this is investigated in connection with the BK inference algorithms where independent modules (called “clusters”) are exploited to speed-up inference or to approximate results.

References