Compiling Dyanamic Fault Trees into Dynamic Bayesian Nets for Reliability Analysis: the RADYBAN Tool[*]

Luigi Portinale, Andrea Bobbio, Daniele Codetta Raiteri, Stefania Montani
Dipartimento di Informatica, Università del Piemonte Orientale
Via Bellini 25g, 15100 Alessandria (ITALY)
email:{portinal,bobbio,raiteri,stefania}@di.unipmn.it

Abstract

In this paper, we present RADYBAN (Reliability Analysis with DYnamic BAyesian Networks), a software tool which allows to analyze systems modeled by means of Dynamic Fault Trees (DFT), by relying on automatic conversion into Dynamic Bayesian Networks (DBN). The tools aims at providing a familiar interface to reliability engineers, by allowing them to model the system to be analyzed with quite a standard formalism (i.e. DFT) based on specific extensions to the well-known methodology of Fault Trees; however, the tool also implements a modular algorithm for automatically translating a DFT into the corresponding DBN, without any explicit intervention from the end user. In fact, when the computation of specific reliability measures is requested, the tool exploits classical algorithms for the inference on Dynamic Bayesian Networks, in order to compute the requested parameters. This is performed in a totally transparent way to the user, who could in principle be completely unaware of the underlying Bayesian Network. However, the use of DBNs allows the tool to be able to compute measures that are not directly computable from DFTs, but that are naturally obtainable from DBN inference. 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.

[*]This work has been partially supported by the EU-Project Crutial IST-2004-27513.

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, 3, 12, 21, 22]. 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 [14], we have shown how BNs can provide a unified framework in which also *Dynamic Fault Trees* (DFT) [6], a rather recent extension to FTs able to treat several types of dependencies, can be represented. However, while reliability engineers are quite familiar with *FT*-based formalisms, they are not usually comfortable with the use of formalisms like *BN* and their extensions. This is also due to the fact that, for reliability purposes, simple and modular techniques are often sufficient for the definition of the required analysis framework. Of course, a clear trade-off exists between the simplicity of the formalism and its modeling, as well as analysis capabilities. FTs are maybe the most simple combinatorial formalism in reliability analysis, but they fail in capturing important aspects like several kind of dependencies among the system components [20, 1, 12]. DFTs overcome some of the limitations of standard FTs, by allowing some kind of dynamic dependencies among components, while still offering a quite simple and structured framework very useful for modeling purposes, but still quite limited from the analysis point of view.

The starting point of our work is to make available to reliability engineers a tool where they can take advantage of the simplicity and modularity of either plain FTs or DFTs, by making them available at the same time a more powerful analysis engine based on *Dynamic Bayesian Networks* (DBN). In fact, the quantitative analysis of DFTs typically requires to expand the model in its whole state space, and to solve the corresponding Continuous Time Markov Chain (CTMC) [6]. Our approach is based on a translation of the DFT into an equivalent DBN. With respect to CTMC, the use of a DBN allows one to take advantage of the factorization in the temporal probability model, via the conditional independence assumptions represented in the DBN.

The tool we have implemented is called RADYBAN (Reliability Analysis with DYnamic BAyesian Networks) and allows the design of the reliability model through a graphical interface where the analyst can access and exploit all the familiar modeling constructs of DFTs; the resulting model is then compiled into an equivalent DBN and the analysis is performed in a transparent way to user, who has just to specify the desired type of analysis algorithm.

2 Dynamic Fault Trees

Fault Trees allow one to represent the combination of elementary causes that lead to the occurrence of an undesired catastrophic event named the *Top Event* (*TE*) [1, 12]. By specifying failure probabilities on the basic components of the modeled system (the elementary causes of the *TE*, also called *basic events*), the whole system unreliability (probability of the *TE*) at a given mission time can be computed.

In recent years, an effort has been documented in the literature, aimed at increasing the modeling power of *FT* by including new primitive gates, able to accommodate complex kinds dependencies. This augmented *FT* language is referred to by the authors as *Dynamic FT* [6, 13]. *DFT* introduce four basic (dynamic) gates: the warm spare (*WSP*), the sequence enforcing (*SEQ*), the probabilistic dependency (*PDEP*) 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. 2(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 λ the failure rate of a powered spare, the failure rate of the unpowered spare is $\alpha\lambda$, with $0 \leq \alpha \leq 1$ called the dormancy factor. Spares are more properly called "hot" if $\alpha = 1$ and "cold" if $\alpha = 0$.

A SEQ gate forces its inputs to fail in a particular order: when a SEQ is found in a DFT, it never happens

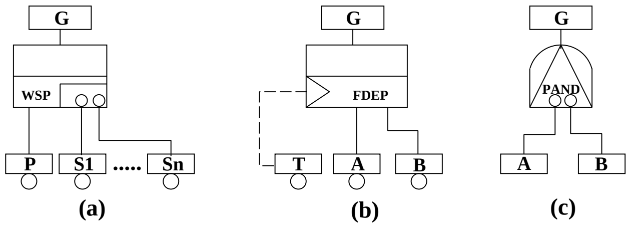


Figure 1: Dynamic gates in a DFT.

that the failure sequence takes place in different orders. SEQ gates can be modeled as a special case of a cold spare [13], so they will not be considered any more in the following[1].

In the FDEP gate (Fig. 2(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 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. 2(c) a failure occurs iff A fails before B, but B may fail before A without producing a failure in G).

3 Dynamic Bayesian Networks

DBNs [5, 18, 16] extend the BN formalism by providing an explicit discrete temporal dimension. The standard *DBN* representation model adopts a discrete time approach, where several time slices are explicited, togheter with information about transitions from a time slice to the next ones. When the Markov assump-

[1]The conceptual difference between the two kind 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.

tion holds (and in particular when we are dealing with a first order Markov process) the future slice at time $t + \Delta$ (Δ being the so called discretization step usually assumed to be 1) is conditionally independent of the past ones given the present slice at time t [11]. In this case, it is sufficient to represent two consecutive time slices called the *anterior* and the *ulterior* layer. The above model of a *DBN* is usually called 2TBN (two time-slice Temporal Bayesian Network)[16, 4]. A DBN (2TBN) is in *canonical form* if the anterior layer contains only variables having influence on the same variable or on another variable at the ulterior level. 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 connect variable X_i^t to variable $X_i^{t+\Delta}$ (being X_i^t the copy of variable X_i at time t). Radyban explicitly uses the notion of temporal arc in its representation.

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 $y_{a:b}$ any stream of observation from time point a to time point b (i.e. a set of instantiated variables Y_i^j with $a \leq j \leq b$). The following tasks can be performed over a DBN:

- **Prediction**: computing $P(X^{t+h}|y_{1:t})$ for some horizon $h > 0$, i.e. predicting a future state taking into consideration the observation up to now; this task is called *filtering* or *monitoring* if $h = 0$.

- **Smoothing**: computing $P(X^{t-l}|y_{1:t})$ 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 or approximate can be exploited in order to implement the above tasks. In the Radyban tool, the user can select either the filtering/prediction or the smoothing task, and for each given task she/he may choose between using a *Junction Tree* (JT) inference [10, 16] or the *Boyen-Koller* (BK) algorithm [4], a parameterized inference algorithm that, depending on the parameters provided (disjoint sets of variables called *clusters*), may return exact as well as approximate results. Such algorithms have been implemented by resorting to Intel PNL (Probabilistic Networks Library), a set of open-source C++ libraries, (`http://www.intel.com/research/mrl/pnl`), to which we have provided some minor adjustments.

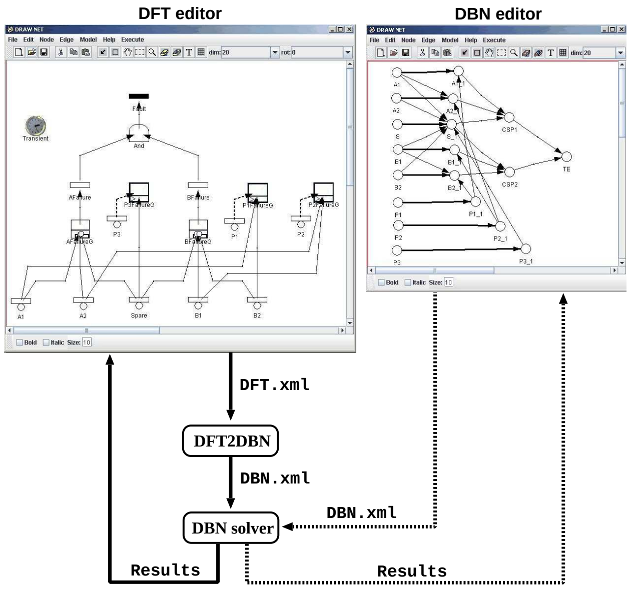


Figure 2: The Radyban tool architecture.

4 The Radyban tool

4.1 Tool functionalities

The main features of Radyban allow the user to: (1) edit a dynamic fault tree and (2) automatically compile a DFT into the corresponding DBN, on which both predictive and diagnostic inference can then be drawn. This is the modality designed for reliability engineers who are not familiar with *BN*-based formalism and who want to take advantage of a *DFT*-based model interface augmented with the whole analysis power of DBNs. However, the tool also allows the possibility of directly editing a (D)BN and then draw inferences on it; of course this modality has to be followed only by users having the necessary background on (D)BNs[2]. The tool architecture is depicted in Fig. 2.

4.2 Graphical interface description

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.

[2]At the current stage, the graphical interface does not allow the user to access the DBN produced by the compilation of a *DFT* (that can however be edited in XML form); we are currently working in order to allow this possibility as well.

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 (allowed by our tool) 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. Another extension on which we are working on is to allow the possibility of the so called *Repair Box*, i.e. a gate modeling the repair (through a suitable repair rate) of components: at the current stage only the repair of basic events (i.e. elementary system components) is allowed.

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* Δ. 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 λ_C: given a discretization step Δ, we can characterize the failure probability of C as

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

In terms of the corresponding DBN, Δ represents the amount of time separating the anterior layer from the ulterior layer. This differs from the model usually adopted in the reliability analysis of a *DFT* which is usually a *CTMC*. The results provided by a *CTMC* are in fact slightly different. As a matter of fact, the two models are not exactly equivalent, since in a *CTMC* transitions occur in a continuous fashion.

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 $fault/hour$ and we set a mission time of T hours, a discretization step $\Delta = 1h$ will require analysis up to step $t = T$, while a discretization step $\Delta = 10h$ 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).

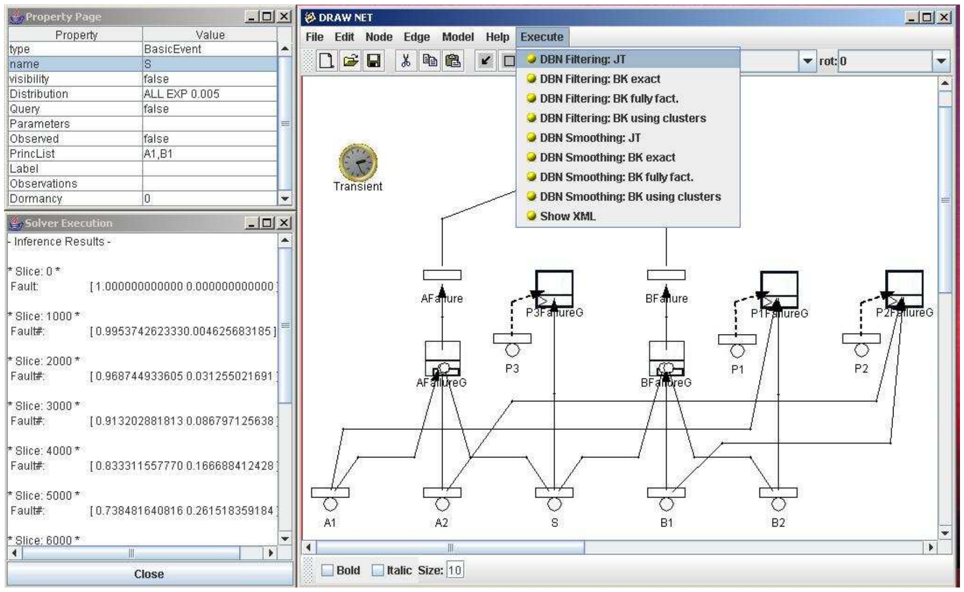
Fig. 3 shows a screenshot of the graphical interface of our tool. It is mainly composed by three windows; the `Main` window allows the user to draw the DFT model, while in the window named `Property Page`, it is possible to set the attributes of the node currently selected in the main window. From the `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`.

4.3 Compiling DFT into DBN

The gates of a *DFT* can be individually compiled into corresponding DBN fragments (see [1, 14] for the technical details); however, each single fragment has then to be combined with each others, in order to produce the compiled DBN corresponding to the input *DFT*.

From the structural point of view, combining different fragments is trivial; just overlaps nodes corresponding to the same variable in different fragments. Fig. 4(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 DBN fragment is shown in the lower part of Fig. 4(a), while the PDEP DBN fragment is shown in the upper part of Fig. 4(a).

While the structural combination is relatively simple (the common part is the one concerning temporal copies of variable B), 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" [9, 17]. The main point refers to the possibility of avoiding 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.



Property Page

Property	Value
type	BasicEvent
name	S
visibility	false
Distribution	ALL EXP 0.005
Query	false
Parameters	
Observed	false
PrincList	A1,B1
Label	
Observations	
Dormancy	0

Solver Execution

```
- Inference Results -

* Slice: 0 *
 Fault:          [ 1.000000000000 0.000000000000 ]

* Slice: 1000 *
 Fault#:         [ 0.9953742623330.004625683185 ]

* Slice: 2000 *
 Fault#:         [ 0.968744933605 0.031255021691 ]

* Slice: 3000 *
 Fault#:         [ 0.913202881813 0.086797125638 ]

* Slice: 4000 *
 Fault#:         [ 0.833311557770 0.166688412428 ]

* Slice: 5000 *
 Fault#:         [ 0.738481640816 0.261518359184 ]

* Slice: 6000 *
```

Figure 3: A screenshot of the RADYBAN tool.

This can be explained by considering a structural transformation called *divorcing* [10]; 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. 4(b), where parent nodes of node $B(t+\Delta)$ in Fig. 4(a) are "divorced" by creating new parents $B'(t+\Delta)$ and $B"(t+\Delta)$. Conceptually, node $B'(t+\Delta)$ represents the combination of nodes $B(t)$ and $T(t+\Delta)$, while node $B"(t+\Delta)$ is the combination of $B(t)$ and $P(t)$. A way of implementing this consists in setting 4 values for nodes $B'(t+\Delta)$ and $B"(t+\Delta)$ such that for example $B'(t+\Delta)$ = "00" iff $B(t) = 0 \wedge T(t+\Delta) = 0$, $B'(t+\Delta)$ = "01" iff $B(t) = 0 \wedge T(t+\Delta) = 1$ and so on. Node $B_f(t+\Delta)$ 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) [10, 19].

For instance, let us suppose that in the example of Fig. 4(a), we quantify $P[B=1|T=1] = p_d = 0.8$, $\alpha = 0.5$, $\lambda = 0.1$, $e^{-\alpha\lambda\Delta} \approx 1-\alpha\lambda$ and $e^{-\lambda\Delta} \approx 1-\lambda$, in the hypothesis that the failure rate is sufficiently

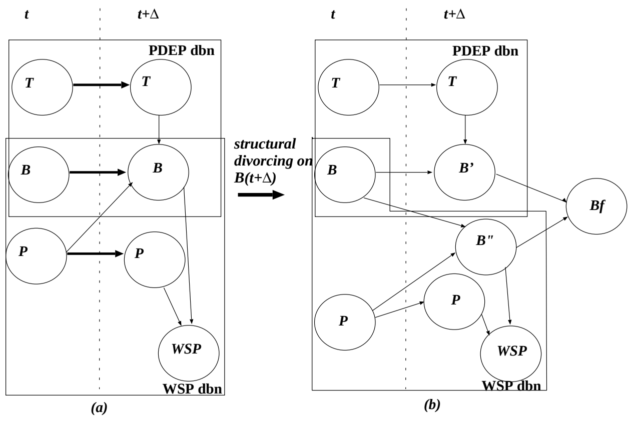


Figure 4: The DBN for a PDEP triggering the spare of a WSP.

small [22] and $\Delta = 1$ (as usual 0 = *working* and 1 = *failed*). With a noisy-OR interaction we could compute for example

$$P[B(t+\Delta)=1|B(t)=0, T(t+\Delta)=1, P(t)=1] = \\ P[B_f(t+\Delta)=1|B'(t+\Delta)=\text{"}01\text{"}, B"(t+\Delta)=\text{"}01\text{"})] = \\ 1-((1-p_d)(1-\lambda)) = 1-0.2\,0.9 = 0.82 \quad (1)$$

It is worth remarking that, in order to compute the CPT produced by combining different fragments on common variables, there is no need to make explicit the divorcing structure of Fig. 4(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. 4(a). In the

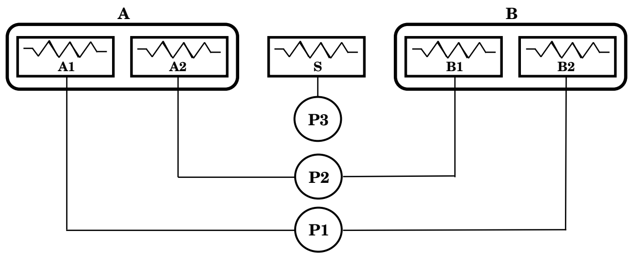


Figure 5: The block scheme of the AHRS'sarchitecture.

current example, our tool will in fact directly produce the structure of Fig. 4(a)[3].

5 An example

The example we report is inspired from [3] and represents an Active Heat Rejection System (AHRS). The block scheme of the AHRS's architecture is depicted in Fig. 5; such system is composed by two redundant thermal rejection units A and B, each one possessing a primary component ($A1$ and $B1$ respectively) and a cold spare ($A2$ and $B2$ respectively). $A1$ and $B2$ are powered by a common source $P1$, which acts as a trigger in a FDEP gate, in which $A1$ and $B2$ are the dependent components. Similarly, $B1$ and $A2$ are powered by $P2$. 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 $P3$. The time to fail of any component in the system is a random variable ruled by the negative exponential distribution; Tab. 1 shows the exponential failure rate of every component.

Fig. 6 shows the DFT for the AHRS system. The DBN, in canonical form, corresponding to the DFT in Fig. 6, is shown in Fig. 7 and is automatically generated given the DFT model, by using our tool.

After the conversion of the DFT in a DBN, we can perform the analysis of the latter by means of our tool. Tab. 2 shows the unreliability of the system versus the mission time varying between 0 and 100 hours, with different discretization steps ($\Delta = 1h, 0.5h, 0.05h$ 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

[3]At the current stage the tool implements, in addition to noisy-OR, another kind of interaction called MSP (Most Severe Prevailing) corresponding to the situation where, given a set of potential causes of an effect, the most severe cause prevails over the others (see [15] for more details).

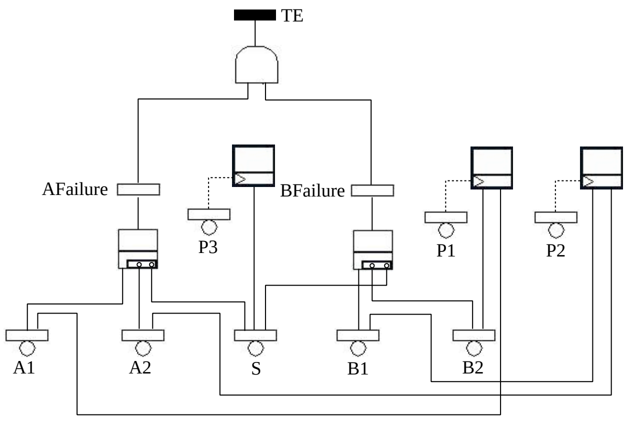


Figure 6: The DFT model of AHRS.

verified by comparison with the results returned by other tools on the same DFT model. Such tools are *DRPFTproc* [2] (based on modularization [8] and conversion to Stochastic Petri Nets of dynamic gates) and *Galileo* [7] (based on modularization, Binary Decision Diagrams (BDD) and CTMCs). Last two columns of tab. 2 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.

Component	failure rate (λ)
A1	0.001 h^{-1}
A2	0.005 h^{-1}
B1	0.002 h^{-1}
B2	0.0035 h^{-1}
S	0.005 h^{-1}
P1, P2, P3	0.003 h^{-1}

Table 1: The failure rates in the AHRS example.

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 $t = 10h$ and $t = 20h$, while it was observed to be failed ($TE =$ `true`) at $t = 60h$. By applying a smoothing algorithm, RADYBAN was able to provide the probabilities of failure of the system in the time

time	$\Delta = 1h$	$\Delta = 0.5h$	$\Delta = 0.05h$	*DRPFTproc*	*Galileo*
$10h$	0.000038	0.000040	0.000041	0.000041	0.000041
$20h$	0.000312	0.000317	0.000321	0.000322	0.000321
$30h$	0.001038	0.001048	0.001057	0.001058	0.001058
$40h$	0.002405	0.002423	0.002438	0.002440	0.002440
$50h$	0.004575	0.004601	0.004625	0.004628	0.004628
$60h$	0.007679	0.007716	0.007749	0.007753	0.007753
$70h$	0.011820	0.011868	0.011911	0.011917	0.011917
$80h$	0.017072	0.017133	0.017188	0.017195	0.017195
$90h$	0.023487	0.023561	0.023627	0.023635	0.023635
$100h$	0.031090	0.031177	0.031251	0.031265	0.031265

Table 2: The unreliability results obtained by RADYBAN at different discretization steps and by other tools.

span $20h \leq t \leq 60h$ (see table 3). For example, we can state that, by knowing that the system was certainly operational at $t = 20h$ and was certainly failed at $t = 60h$, the probability that it was already failed at $t = 50h$ is about 0.4%. This suggest that a very unlikely event has occurred, because just 10 hours before the observation of the failure, the system was almost definitely operational.

time	RADYBAN unreliability
$10h$	0.000000
$20h$	0.000000
$30h$	0.000736
$40h$	0.002118
$50h$	0.004305
$60h$	1.000000

Table 3: Smoothing results.

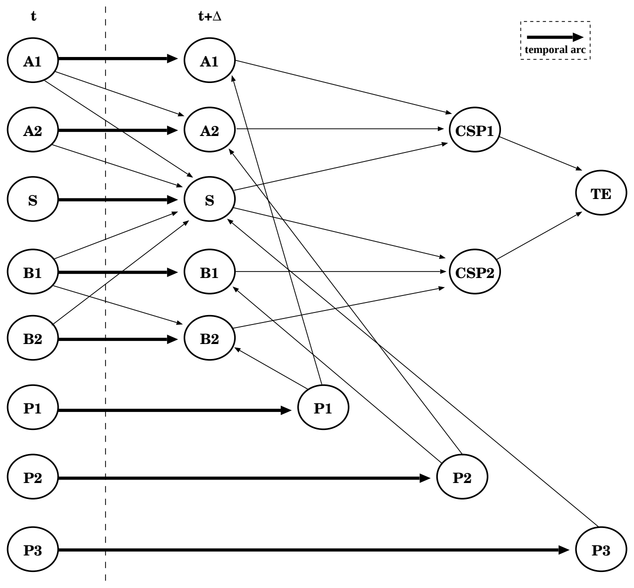


Figure 7: The DBN corresponding to the DFT in Fig. 6.

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.

6 Conclusions

In this paper, we have described RADYBAN, a tool that allows reliability engineers to work ar different modeling level, while still having available all the inference power of the DBN formalism. The tool is aimed at helping reliability engineers in modeling the system to be analyzed via a standard formalism like *DFT*, by making them available at the same time all the inference power of DBNs.

We feel that the approch implemented in the tool can be a step forward in making available formalisms based on Bayesian nets to the reliability community, without asking reliability practitioners to renounce to their familiar constructs like those used in *FT* or *DFT*. 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 complex repair policies and the specification of conditional dependencies among basic events.

References

[1] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla. Improving the analysis of dependable systems by mapping fault trees into bayesian networks. *Reliability Engineering and System Safety*, 71:249–260, 2001.

[2] A. Bobbio and D. Codetta Raiteri. Parametric fault-trees with dynamic gates and repair boxes. In *Proceedings Reliability and Maintainability Symposium RAMS2004*, pages 101–106, Los Angeles, USA, 2004.

[3] H. Boudali and J. Bechta-Dugan. A new bayesian network approach to solve dynamic fault trees. In *Proceedings Reliability and Maintainability Symposium RAMS2005*, pages 451–456, 2005.

[4] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings UAI 1988*, pages 33–42, 1998.

[5] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[6] J. Bechta Dugan, S.J. Bavuso, and M.A. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41:363–377, 1992.

[7] J. Bechta Dugan, K.J. Sullivan, and D. Coppit. Developing a low-cost high-quality software tool for dynamic fault-tree analysis. *IEEE Transactions on Reliability*, 49(1):49–59, 2000.

[8] R. Gulati and J. Bechta-Dugan. A modular approach for analyzing static and dynamic fault trees. In *Proceedings Reliability and Maintainability Symposium RAMS1997*, pages 1–7, 1997.

[9] D. Heckerman and J.S. Breese. Causal independence for probability assessment and inference using bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics*, 26(6):826–831, 1996.

[10] F.V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

[11] U. Kjaerulff. dhugin: a computational system for dynamic time-sliced bayesian networks. *International Journal of Forecasting*, 11:89–101, 1995.

[12] H. Langseth and L. Portinale. Bayesian networks in reliability. *Reliability Engineering and System Safety*, 92(1):92–108, 2007.

[13] R. Manian, D.W. Coppit, K.J. Sullivan, and J.B. Dugan. Bridging the gap between systems and dynamic fault tree models. In *Proceedings IEEE Annual Reliability and Maintainability Symposium*, pages 105–111. IEEE Computer Society Press, Washington, DC, 1999.

[14] S. Montani, L. Portinale, and A. Bobbio. Dynamic bayesian networks for modeling advanced fault tree features in dependability analysis. In *Proc. ESREL 2005, Tri City*, pages 1414–1422, 2005.

[15] S. Montani, L. Portinale, A. Bobbio, and D. Codetta-Raiteri. RADYBAN: a tool for reliability analysis of dynamic fault trees through conversion into dynamic bayesian networks. *Reliability Engineering and System Safety*, in press, 2007.

[16] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD Thesis, UC Berkley, 2002.

[17] D. Poole N.L. Zhang. Exploiting causal independence in Bayesian network inference. *Journal of Artifical Intelligence Research*, 5:301–328, 1996.

[18] P.Dagum, A. Galper, and E. Horwitz. Dynamic network models for forecasting. In *Proc. UAI'92*, pages 41–48, 1992.

[19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1989.

[20] L. Portinale and A. Bobbio. Bayesian networks for dependability analysis: an application to digital control reliability. In *15-th Conference Uncertainty in Artificial Intelligence, UAI-99*, pages 551–558, 1999.

[21] J.G. Torres-Toledano and L.E. Sucar. Bayesian networks for reliability analysis of complex systems. In *Lecture Notes in Artificial Intelligence*, volume 1484, pages 195–206. Springer Verlag, Berlin, 1998.

[22] P. Weber and L. Jouffe. Reliability modelling with dynamic bayesian networks. In *SafeProcess 2003, 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Washington DC, 2003.