A Continuous-Time Bayesian Network Reliability Modeling, and Analysis Framework

Hichem Boudali, *Member, IEEE,* and Joanne Bechta Dugan, *Fellow, IEEE*

***Abstract*—We present a continuous-time Bayesian network (CTBN) framework for dynamic systems reliability modeling and analysis. Dynamic systems exhibit complex behaviors and interactions between their components; where not only the combination of failure events matters, but so does the sequence ordering of the failures. Similar to dynamic fault trees, the CTBN framework defines a set of 'basic' BN constructs that capture well-defined system components' behaviors and interactions. Combining, in a structured way, the various 'basic' Bayesian network constructs enables the user to construct, in a modular and hierarchical fashion, the system model. Within the CTBN framework, one can perform various analyses, including reliability, sensitivity, and uncertainty analyses. All the analyses allow the user to obtain closed-form solutions.**

***Index Terms*—Bayesian networks, dynamic systems, reliability modeling and analysis.**

ACRONYMS[1]

BN	Bayesian network
CPD	conditional probability distribution
CSP	cold spare
CTBN	continuous-time Bayesian network
DBN	dynamic Bayesian network
DFT	dynamic fault tree
DTBN	discrete-time Bayesian network
FDEP	functional dependency
FT	fault tree
HSP	hot spare
MPD	marginal probability distribution
MTTF	mean time to failure
PAND	priority AND
PDF	probability density function
RBD	reliability block diagram
RV	random variable
SEQ	sequence enforcing
WSP	warm spare

NOTATION

α	dormancy factor
$\delta(t)$	impulse function
$\lambda_1, \lambda_2, \beta, \beta_L, \beta_H$	failure rates
f_X	marginal probability density function of variable X
$f_{X\|Y}$	conditional probability density function of variable X given Y
$f_{X,Y}$	joint probability density function of variables X and Y
F	cumulative distribution function of PDF f
$pa(X_i)$	the set of the parent variables of node X_i
$u(t)$	Heaviside unit-step function
x	a specific value taken by random variable X
X	a random variable

Manuscript received February 12, 2005; revised May 14, 2005 and June 14, 2005. Associate Editor: M. Xie.

The authors are with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904-4743 USA (e-mail: hboudali@virginia.edu; jbd@virginia.edu).

Digital Object Identifier 10.1109/TR.2005.859228

[1]The singular and plural of an acronym are always spelled the same.

I. INTRODUCTION

TODAY'S reliability methodologies and tools need to assess increasingly large, complex systems. When developing such methodologies and tools, there are several aspects we need to be aware of: (1) The modeling power, (2) the ease in specifying a model, and (3) the computational efficiency.

Complex system components exhibit dynamic behavior, where not only the combination of failing components reflects the state of the system, but also the sequence in which these components fail. Conventional combinatorial reliability assessment methods (e.g., static FT, RBD) fail to capture such dynamic behavior. Markov chains have proved to be a versatile formalism for modeling dynamic systems. However, they present some limitations: (1) Markov chains are a low level modeling formalism, and manually specifying a Markov chain for a large system becomes a cumbersome, error-prone task. For this reason, Markov chains are usually automatically derived from a high level modeling description language (e.g., DFT in the Galileo tool [1]). (2) Markov chain modeling is limited to Markov processes, which generally requires all state holding times to be exponentially distributed. (3) Markov chains are faced with the infamous state space explosion problem; in fact, the number of states grows exponentially with the size of the system (i.e., number of components in the system). Consequently, the number of differential equations to be solved simultaneously grows exponentially with the size of the system. The state space explosion is one of the main limitations in using Markov chains for modeling large systems.

A software tool for reliability analysis ought to be easy and intuitive to use. Engineers generally view the system in terms of a collection of components interacting with each other. Higher level formalisms tend to provide such a view and abstraction of the system. For instance, FT and DFT map each system component to an event in the tree; and express the component interactions through the use of specialized gates (also called constructs). Each gate describes a particular component behavior or a particular interaction between a set of components (e.g., the WSP gate in Galileo is used to describe a situation where a primary working unit possesses a backup spare unit which is turned on once the primary unit fails). In the Galileo DFT reliability tool, there are several of these predefined specialized gates which effectively form a library. A real system is generally broken down into modules because a modularized model of the system is indeed easier to specify, understand, and solve. Reliability formalisms need to capture this modular nature of systems, and enable the user to specify the system model in a modular fashion. RBD, static FT, and DFT are inherently modular because they map system components to events in their diagram or tree.

Any solution of a large system model is faced with the problem of long computational times. Modularizing the system helps break down the size of the system into smaller, more manageable modules, and thus can reduce the overall computational time. However, in many circumstances, the size of a single module remains significant, potentially leading to an unreasonably long computational time. In Markov chain modeling, long computation times arise due to a state space explosion of the Markov chain. There exist some techniques (e.g., states lumping, and states truncation) to reduce the size of the Markov chain; however, the applicability of these techniques is not always evident, and/or may lead to gross approximations.

The objective of this work is to find an alternative formalism for modeling and analyzing large dynamic systems, and address the problems and issues mentioned above. We propose a temporal Bayesian network reliability modeling and analysis formalism.

II. Background

In this section, we provide a brief background on DFT, and BN; and their usage in reliability assessment. The remainder of the paper is divided as follows. In Section III, we present our continuous-time BN formalism, and show, as examples, the AND gate and the WSP gate CTBN. In Section IV, we illustrate the usage of the CTBN formalism through a hypothetical example system, and show the different types of analyses that can be performed. Section V provides a discussion, and some concluding remarks.

A. Dynamic Fault Trees

Dynamic fault trees extend traditional fault trees to handle failure sequence (or temporal), and functional dependencies. Traditional FT, also called standard or static FT, are combinatorial models. A combinatorial model only captures the combination of events, and not the sequential ordering of their occurrences. Combinatorial models become, therefore, inadequate to model today's complex dynamic systems. Dynamic fault trees define special gates that capture a variety of failure sequence and functional dependencies.

The Galileo tool currently uses (in addition to the static gates AND, OR, and K/M) six dynamic gates: The functional dependency gate (FDEP), the spare gates HSP (for hot spares), CSP (for cold spares), WSP (for warm spares), the priority AND gate (PAND), and the sequence enforcing gate (SEQ). The Galileo user models a system as a fault tree using static and dynamic gates. A modularization algorithm [2] finds the different modules composing the system. Static modules are solved using a binary decision diagram based algorithm, and dynamic modules are solved using a Markov chain. Each dynamic module is in fact automatically, and transparently to the user, converted into an equivalent Markov chain. The modules are solved separately, and their solutions are then combined to get the overall system failure probability. The reader is encouraged to consult references [3]–[5] for a thorough description of dynamic gates.

B. Bayesian Networks

A Bayesian network[2] is a directed acyclic graph comprised of nodes and arcs. Nodes[3] represent random variables (RV), and directed arcs between pairs of nodes represent dependencies between the RV. A Bayesian network uniquely defines a joint probability distribution over all the RV present in the graph [6], [7]. Any probabilistic query (e.g., probability of X, and Y to be in state x, and y respectively, given that variable Z is in state z) can be answered once the joint probability distribution over all the RV is known. The concept of *d-separation* [6]–[8] restricts relevance between RV, and allows a compact representation of the joint probability distribution. In fact, by examining a Bayesian network structure (or graph), one can identify the (conditional) independence assumptions between the various random variables constituting the BN. It is this very particular characteristic of BN that makes the usage of BN to model large systems appealing. Indeed, a Markov chain model is a *global-state* model, where each of the states in the Markov chain explicitly describes the state of all the system variables. It's this very particularity that makes Markov chains vulnerable to the state space explosion problem. On the other hand, in a BN, each node A is only affected by a limited number of nodes that form A's *Markov blanket*. The Markov blanket of node A is the set consisting of the parents of A, the children of A, and the nodes sharing a child with A [7], [9]. Given the states of the nodes belonging to A's Markov blanket, A is d-separated from the rest of the network. This property naturally breaks down the complexity of the system model by avoiding a global state representation. In this respect, a BN is a *local-state* model.

In a BN, if there is an arc from node A to node B, we say that A is a parent of B, and B is a child of A. Nodes without incoming arcs, i.e., without parents, are called root nodes; and nodes without outgoing arcs, i.e., without children, are called leaf nodes. If there exists a directed path from node A to node B; then B is a descendant of A, and A is an ancestor of B. A RV can be in any one of a number of mutually exclusive states, and can be continuous or discrete. Each root node has a marginal probability distribution (MPD) associated with it, and all other nodes have conditional (conditioned on the state of the parent nodes) probability distributions associated with them.

The conditional probability distribution (CPD) of a node X quantifies the effects of the parent nodes upon X. The CPD of a RV X specifies, for each of the states of X, the probability

[2]A Bayesian network is also known as Bayesian belief network, causal network, causal net, graphical probability network, probabilistic cause/effect model, or probabilistic network.

[3]We will interchangeably use the words node, and variable.

of being in a particular state conditioned on the states of each of its parent nodes. The joint probability distribution is determined using the *chain rule*, and assuming the conditional independence assumptions, encoded in the BN structure, between the variables. The joint probability distribution of a set of variables $X_1, X_2, \ldots, X_n$ is [8]

$$f_{X_1,X_2,\ldots,X_n}(x_1,x_2,\ldots,x_n) = \prod_{i=1}^{n} f_{X_i|pa(X_i)}\left(x_i|pa(X_i)\right) \tag{1}$$

where $pa(X_i)$ denotes the set of all the parent variables of node X_i.

The MPD of a particular RV is found by *marginalizing* the joint probability distribution with respect to the RV. For example, assume we have three continuous nonnegative real variables A, B, and C; i.e., each variable belongs to $[0,+\infty)$. The joint probability distribution (expressed as a density function) $f_{A,B,C}$ is known. The MPD f_B of B, for instance, can be obtained by marginalizing with respect to A and C

$$f_B(b) = \int_0^\infty \int_0^\infty f_{A,B,C}(a,b,c)dadc$$

Temporal Bayesian Networks: To account for temporal (or sequence) dependencies, we need an explicit representation of time in a BN. Temporal BN can be divided into two broad categories according to their time representation: *Instant-based* approaches (also called *time-sliced* approaches), and *interval-based* approaches (also called *event-based* approaches).

In *instant-based* approaches, time is divided into successive time instants, and a RV (i.e., a node in the BN) is associated with each time instant. The BN model is essentially obtained by generating a BN for a specific time instant, and repeating the same structure for all other time instants, thus covering the whole time range of interest. Arcs are then also added between RV belonging to different time instants. Dynamic Bayesian networks (DBN) [10], [11] are an instance of widely used, popular instant-based temporal BN.

In *interval-based* approaches, the time line is subdivided into a finite number of disjoint time intervals. A temporal RV represents an event that can have a certain outcome, and that can take place at a certain time interval; thus, a state of the RV is defined as being the event's outcome, and the time interval at which the event took place. An interval-based temporal BN is generally used to model irreversible processes.[4] Temporal nodes Bayesian networks (TNBN) [12], net of irreversible events in discrete time (NIEDT) [13], and networks of probabilistic events in discrete time (NPEDT) [14] are all examples of interval-based temporal BN.

Recent work by Nodelman *et al.* has looked at a new way to express (homogenous) Markov processes using a Bayesian network approach. In [15], the authors present a language to describe Markov processes that evolve over continuous time. The idea is to define a set of *conditional Markov processes*, one process per variable, and then to compose those processes to get the overall system Markov process. Defining a conditional Markov process is similar to defining a CPD in a BN. A conditional Markov process is fully characterized by its conditional intensity matrix. Although this work is different from ours, the authors use the same acronym (i.e., CTBN in [15]) to describe their framework.[5]

The Use of BN in Dependability Assessment: Bayesian networks have been extensively used in certain areas such as medical diagnosis, system troubleshooting, and manufacturing control. However, the application and contribution of temporal BN to the area of dependability analysis remains modest. In fact, specifying and using a temporal BN for a specific domain knowledge is a difficult task. The difficulty lies in: (1) defining, especially for large systems, a correct network structure reflecting the 'true' behaviors and interactions between the system, or process, components; and (2) specifying the right values for the prior, and the conditional probability table entries (or density functions). In reference [16], Bobbio *et al.* apply BN to reliability analysis. They provide a mapping between static (AND, OR, K/M gates) FT and BN, and show how inference in the latter can be used to obtain the FT top gate failure probability. They also illustrate how additional benefits can be obtained from the usage of BN at both the modeling and the analysis levels. However, this work has been only carried out for static fault trees, and therefore not applicable for dynamic systems that exhibit complex sequential dependencies between their components. Torres-Toledano and Succar [17] have also conducted similar work where a mapping between RBD and BN has been established. In [18], Bouissou *et al.* use a BN to demonstrate how to perform availability computation on a multi-state system. In Bouissou's framework, the BN probability tables are populated using asymptotic (or stationary) probabilities. The asymptotic probabilities are computed based on the interactions between the components, and using other analysis techniques such as Markov chains. In the dynamic domain, Weber *et al.* [19] show how to use a 2-time-slice DBN to model temporal dependencies between components for reliability calculations. The authors also demonstrate the equivalence between their DBN model, and a Markov chain, i.e., they both possess the Markov property. Thus the model is applicable exclusively to Markovian processes.

In our previous work [20], we defined an interval-based discrete-time BN (DTBN) framework for reliability modeling and analysis of dynamic nonrepairable systems. We also provided the equivalent DTBN for each of the DFT gates present in the Galileo tool, and showed how additional BN constructs can be added to account for new component behaviors and interactions (e.g., deterministic time delay, inhibition). Similar to FT, the DTBN reliability framework provides a high level modeling environment, and is hierarchical and modular in nature. The DTBN framework is expandable from a modeling power point of view because new constructs can be incrementally added to the existing library of constructs. For each new construct, we need to identify the BN structure, showing the various dependencies between the variables (or nodes); and the marginal, or

[4]Events that happen in at most one time interval.

[5]In the remainder of the paper, the acronym CTBN refers to our definition of the continuous time Bayesian network framework.

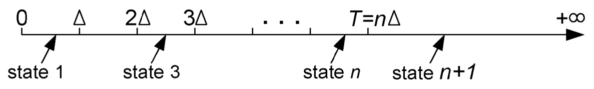

Fig. 1. The correspondence between the variable states, and the failure time intervals.

conditional, probability distribution of each node. From a model specification point of view, the user specifies and combines high level constructs when building a system. The underlying details of the BN are automatically generated, and transparent to the user. The DTBN solution is carried out using a standard BN inference algorithm. The solution is flexible as it provides a tradeoff between accuracy and computation speed depending on the time granularity that the user sets.

A node, or RV, in our reliability BN represents the state of a system component. A component can be either a basic system component, or a subsystem. A subsystem is represented by a gate[6] whose output reflects the state of the subsystem (i.e., working or failed). The RV X represents the *failure event* of basic component X, or gate output X.

In the DTBN formalism, we divide the time line, from time $t = 0$ to time $t = T$ (T is mission time), into n intervals of length Δ each. n is the *time granularity*. Each variable (represented by a node) has $n+1$ states. The n first states represent the failure of component X in one of the first n time intervals (i.e., during the mission time); and the last state $(n+1)$ represents the survival of X for the duration of the mission. Note that to establish a temporal ordering among a set of RV, all RV need to have the same time granularity. The statement "RV X is in state x (i.e., $X = x$)" means that the basic component or the gate fails in time interval x; i.e., $((x-1)\Delta, x\Delta]$. Fig. 1 depicts the correspondence between the variable states and the failure time intervals. For further details on the DTBN formalism, application examples, and performing various analyses on the system BN, refer to [20], [21]. At the limit (i.e., as n tends to infinity) the DTBN becomes a CTBN. Next, we define the CTBN.

III. The Continuous-Time BN Formalism

The same framework is used whether the BN variables are discrete or continuous. In fact, the underlying BN structure used in the CTBN is the same as the one used in the DTBN. The difference is that, in a CTBN, variables are continuous, (conditional) probabilities are expressed in terms of (conditional) probability density functions, and the joint probability distribution is expressed as a joint probability density function. Actually, a DTBN is an approximation of the CTBN, where time is discretized. The (conditional) probabilities for each of the nodes, in a DTBN, are derived from the (conditional) probability density functions defined in the continuous domain.

Similar to the DTBN formalism, the CTBN formalism is hierarchical and modular from both a modeling point of view, and an analysis point of view. The two main advantages for using a CTBN over a DTBN are

- Memory savings: Because the (conditional) probabilities are expressed as parametric functions, there are no multi-dimensional tables, as with DTBN, to be stored.
- Closed-form solution: An exact closed-form analytical expression of the system reliability is derived.

As mentioned earlier, the DTBN solution is carried out using a standard BN inference algorithm. Unfortunately, there isn't a theory for exact BN inference in continuous-time BN with general distributions. However, a theory for exact inference exists where the distributions are Gaussians [22]; and recently, Moral [23] also described a theory for exact inference where distributions are specified as a mixture of truncated exponentials. At this point, the CTBN exact closed-form solution is obtained by simply executing a series of symbolic integrations.

In the CTBN formalism, the variables have a continuous state space. The state space is the failure time of the system component. The statement "random variable X is in state x" means that the system component represented by X has failed in the time instant x, with x belonging to the set of *nonnegative real* numbers. Table I provides a comparison between the DTBN, and the CTBN formalisms.

To capture the kind of (temporal) behaviors and dependencies found in complex dynamic systems, we will use two special functions: The *unit-step function* (also called the Heaviside unit-step function), and the *impulse function* (also called the Dirac delta function). In the next section, we define these two functions, and explain their use in our CTBN formalism. We then present the CTBN formalism using the AND and WSP gates as illustrations.

A. The Unit-Step Function

The unit-step function $u(t-\tau)$ is conventionally defined [24] as

Definition 1: The Unit-step function

$$u(t-\tau) = \begin{cases} 0, & \text{if } t < \tau \\ \frac{1}{2}, & \text{if } t = \tau \\ 1, & \text{if } t > \tau \end{cases}$$

where t and τ belong to the set of nonnegative real numbers.

$u(t-\tau)$ is discontinuous at $t = \tau$, where the discontinuity of the step function occurs. In the CTBN formalism, the unit-step function $u(t-\tau)$ is used to denote that an event occurring at time t can not take place before the time instant τ, or equivalently that time τ precedes time t.

B. The Impulse Function

The impulse function is used as a PDF,[7] and expresses the fact that a variable takes on a specific, unique value. The impulse function is defined as [24], [25].

Definition 2: Impulse function

$$\delta(t-\tau) = 0, \qquad for\ t \neq \tau$$

and

$$\int_{-\infty}^{\infty} \delta(t-\tau)dt = 1.$$

[6]Gate (alternatively called construct) is used here as a generic word for describing any subsystem reflecting a certain interaction between a set of system components.

[7]It satisfies the properties of a legitimate PDF.

TABLE I
DTBN AND CTBN COMPARISON

	DTBN	CTBN
State space	$x \in \mathbb{N}$ (discrete)	$t \in \mathbb{R}^+$ (continuous)
State representation	*state* $x \triangleq$ failing in time interval $((x-1)\Delta, x\Delta]$	*state* $t \triangleq$ failing at time instant t
Advantages	Solved using a standard BN inference engine	closed-form solution, memory savings
Disadvantages	approximate solution, high memory needs	No general-distribution BN inference engine

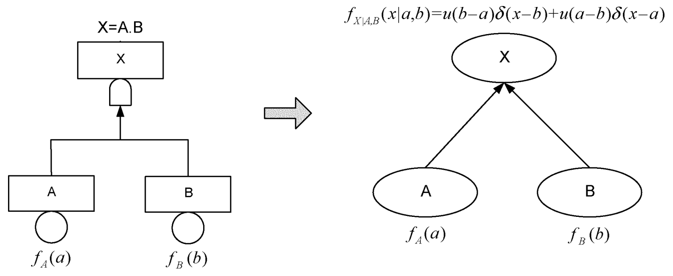

Fig. 2. The fault tree AND gate, and its equivalent CTBN.

Because all the variables in our framework are nonnegative real numbers (i.e., $t, \tau \geq 0$), we rewrite the integral in Definition 2 as

$$\int_{-\infty}^{\infty} \delta(t-\tau)dt = \int_{0}^{\infty} \delta(t-\tau)dt = 1 \tag{2}$$

The following is an important property [24] of the impulse function which will be used extensively.

Property 1:

$$\int_{0}^{\infty} f(t)\delta(t-\tau)dt = f(\tau) \tag{3}$$

The interpretation of the impulse function is as follows: Consider a term $P\delta(t-\tau)$ of a time-to-failure PDF, where P is a fixed number between 0 and 1. $P\delta(t-\tau)$ represents an event failing at exactly time $t=\tau$ with a probability P. When $\tau=\infty$, $P\delta(t-\infty)$ expresses the fact that the event will never happen (i.e., happens at a time in infinity) with a probability P. The PDF term $P\delta(t-\infty)$ is typically used to denote a *defective* failure distribution.

C. The AND Gate CTBN

The fault tree AND gate, and the BN equivalent, are shown in Fig. 2. The structure of the AND gate CTBN is the same as the AND gate DTBN structure. Nodes A and B are root nodes, and each possesses a marginal PDF, $f_A(a)$ and $f_B(b)$ respectively. X is the output of the AND gate. The conditional PDF $f_{X|A,B}(x|a,b)$ of node X is[8]

$$f_{X|A,B}(x|a,b) = u(b-a)\delta(x-b) + u(a-b)\delta(x-a) \tag{4}$$

where the first term $u(b-a)\delta(x-b)$ states that when B fails after A, the state of node X (i.e., the failure time of the gate) is equal to the state of node B (i.e., the time at which B failed). While the second term $u(a-b)\delta(x-a)$ states that, when A fails after B, the state of node X is equal to the state of node A.

Notice that $f_{X|A,B}$ is a proper distribution function; in fact, using Definition 1, and (2)

$$\int_{0}^{\infty} f_{X|A,B}(x|a,b)dx = u(b-a) + u(a-b) = 1 \qquad \forall a,b. \tag{5}$$

The joint failure probability density function of the BN shown on Fig. 2 is

$$f_{ABX}(a,b,x) = f_{X|A,B}(x|a,b) f_B(b) f_A(a). \tag{6}$$

To find the marginal failure PDF of X, we need to integrate (i.e., marginalize), $f_{ABX}(a,b,x)$ with respect to a and b from 0 to ∞. Using[9] (2), and Property 1, we get

$$\begin{aligned} f_X(x) &= \int_0^\infty \int_0^\infty u(b-a)\delta(x-b) f_B(b) f_A(a) db da \\ &\quad + \int_0^\infty \int_0^\infty u(a-b)\delta(x-a) f_B(b) f_A(a) db da \\ &= \int_0^\infty f_B(b)\delta(x-b) db \int_0^b f_A(a) da \\ &\quad + \int_0^\infty f_A(a)\delta(x-a) da \int_0^a f_B(b) db \\ &= \int_0^\infty f_B(b)\delta(x-b) F_A(b) db \\ &\quad + \int_0^\infty f_A(a)\delta(x-a) F_B(a) da \\ &= [F_B(x) F_A(x)]'. \end{aligned} \tag{7}$$

The probability of failure of the gate output X in time interval $[0,t]$ is

$$F(t) = Pr(X \leq t) = \int_0^t f_X(x) dx = F_B(t) F_A(t) \tag{8}$$

[8]Refer to [26] for details.

[9]We also use the fact that $\delta(t-\tau) = \delta(\tau - t)$.

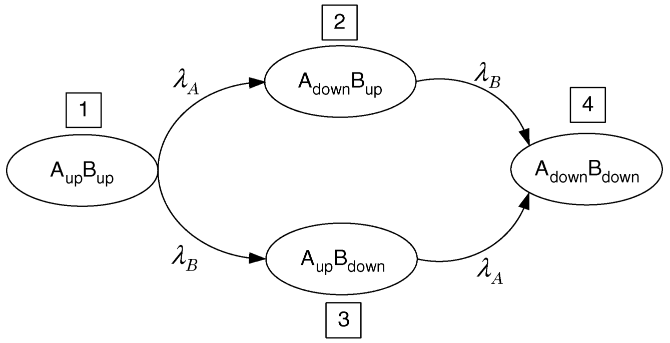

Fig. 3. The corresponding Markov chain of an AND gate.

Equation (8) is the result we expect for the failure probability of the AND gate having s-independent inputs (i.e., $f_{A|B}(a|b) = f_A(a)$, and $f_{B|A}(b|a) = f_B(b)$). As an example, let's assume that A and B are exponentially distributed, i.e., $F_A(a) = 1 - e^{-\lambda_A a}$, and $F_B(b) = 1 - e^{-\lambda_B b}$; then

$$\begin{aligned} Pr(X \le t) &= \int_0^t f_X(x)dx \\ &= F_B(t)F_A(t) \\ &= 1 - e^{-\lambda_A t} - e^{-\lambda_B t} + e^{-(\lambda_A+\lambda_B)t} \end{aligned} \tag{9}$$

The above analytical solution corresponds to the closed-form solution given by the Markov chain in Fig. 3, where the AND failure probability corresponds to the probability of being in state 4.

As a second example, let's assume that the failure distributions of A and B follow a Weibull distribution with shape parameter s, and scale parameter η; i.e., $F_A(a) = 1 - e^{-(a/\eta)^s}$, and $F_B(b) = 1 - e^{-(b/\eta)^s}$. Then, in this case

$$Pr(X \le t) = F_B(t)F_A(t) = 1 - 2e^{-\left(\frac{t}{\eta}\right)^s} + e^{-2\left(\frac{t}{\eta}\right)^s} \tag{10}$$

Note that, in this case, the corresponding Markov chain is nonhomogeneous; i.e., transition rates are time dependent.

D. The WSP Gate CTBN

The WSP gate generalizes the CSP, and the HSP gate. Fig. 4 shows the WSP dynamic fault tree, and its equivalent BN. Node A is the primary unit, and node B is the spare unit. When the system starts functioning, the primary unit A is active, and the spare B is in a standby (or dormant) mode. In standby mode, B's hazard rate is reduced by a factor α, called the dormancy factor. Note, in the BN on Fig. 4, how node B is dependent upon node A (i.e., arc $A \to B$). To formulate B's conditional PDF, the *in-isolation* failure distribution $F_{Bi}(t)$ (or hazard rate function $h_{Bi}(t)$) of B must be known in advance.

The conditional PDF of node B is[10]

$$\begin{aligned} f_{B|A}(b|a) = u(a-b)\alpha f_{Bi}(b)\left[1 - F_{Bi}(b)\right]^{\alpha-1} \\ +u(b-a)f_{Bi}(b-a)\left[1 - F_{Bi}(a)\right]^{\alpha} \end{aligned} \tag{11}$$

In the HSP case (i.e., $\alpha = 1$), (11) reduces to

$$f_{B|A}(b|a) = f_{Bi}(b) \tag{12}$$

[10]A detailed derivation of $f_{B|A}(b|a)$ is provided in the Appendix.

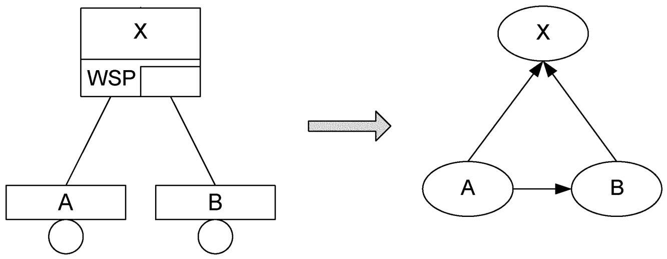

Fig. 4. The WSP fault tree, and the corresponding Bayesian network.

In the CSP case (i.e., $\alpha = 0$), (11) reduces to

$$f_{B|A}(b|a) = u(b-a)f_{Bi}(b-a) \tag{13}$$

The marginal failure PDF $f_B(b)$ is

$$\begin{aligned} f_B(b) &= \int_0^\infty \Big\{u(a-b)\alpha f_{Bi}(b)\left[1 - F_{Bi}(b)\right]^{\alpha-1} \\ &\qquad +u(b-a)f_{Bi}(b-a)\left[1 - F_{Bi}(a)\right]^{\alpha}\Big\} f_A(a)da \\ &= \int_b^\infty \alpha f_{Bi}(b)\left[1 - F_{Bi}(b)\right]^{\alpha-1} f_A(a)da \\ &\quad + \int_0^b f_{Bi}(b-a)\left[1 - F_{Bi}(a)\right]^{\alpha} f_A(a)da \\ &= \alpha f_{Bi}(b)\left[1 - F_{Bi}(b)\right]^{\alpha-1}\left[1 - F_A(b)\right] \\ &\quad + \int_0^b f_{Bi}(b-a)\left[1 - F_{Bi}(a)\right]^{\alpha} f_A(a)da \end{aligned} \tag{14}$$

The second term in (14) is not trivial to find for any general distribution. As an example, assume $f_A(a) = \lambda e^{-\lambda a}$, and $f_{Bi}(b) = \lambda e^{-\lambda b}$. Then

$$\begin{aligned} f_B(b) &= \alpha f_{Bi}(b)\left[1 - F_{Bi}(b)\right]^{\alpha-1}\left[1 - F_A(b)\right] \\ &\quad + \int_0^b f_{Bi}(b-a)\left[1 - F_{Bi}(a)\right]^{\alpha} f_A(a)da \\ &= \lambda\alpha e^{-\lambda(1+\alpha)b} + \frac{\lambda}{\alpha}e^{-\lambda b}[1 - e^{-\lambda\alpha b}] \\ &= \frac{\lambda(\alpha^2-1)}{\alpha}e^{-\lambda(1+\alpha)b} + \frac{\lambda}{\alpha}e^{-\lambda b} \end{aligned} \tag{15}$$

We can now compute $Pr(B \le t)$ as

$$\begin{aligned} Pr(B \le t) &= \int_0^t f_B(b)db \\ &= \int_0^t \left\{\frac{\lambda(\alpha^2-1)}{\alpha}e^{-\lambda(1+\alpha)b} + \frac{\lambda}{\alpha}e^{-\lambda b}\right\} db \\ &= 1 - \frac{e^{-\lambda t}}{\alpha} + \frac{1-\alpha}{\alpha}e^{-\lambda(1+\alpha)t} \end{aligned} \tag{16}$$

The above analytical solution corresponds to the closed-form solution we get from the Markov chain depicted in Fig. 5 (probability of being in state 3 or 4).

Fig. 6 shows the CTBN for the WSP gate. The output X of the WSP is simply an AND gate; i.e., $X = A$ AND B. Due to

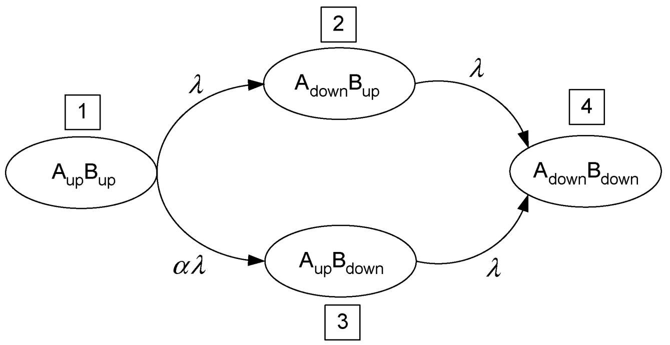

Fig. 5. The corresponding Markov chain of a WSP gate.

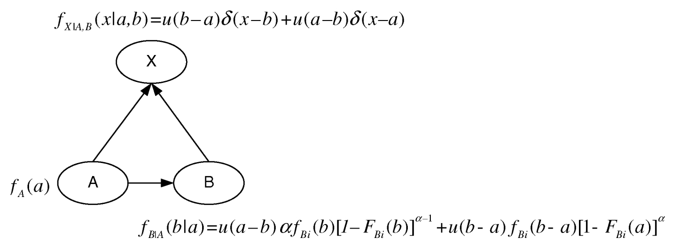
$f_{X|A,B}(x|a,b)=u(b-a)\delta(x-b)+u(a-b)\delta(x-a)$
X
$f_A(a)$
A
B
$f_{B|A}(b|a)=u(a-b)\alpha f_{Bi}(b)[1-F_{Bi}(b)]^{\alpha-1}+u(b-a)f_{Bi}(b-a)[1-F_{Bi}(a)]^{\alpha}$

Fig. 6. The WSP gate CTBN.

space limitations, the more general and complicated configuration of multiple spares shared among multiple WSP gates is not presented in this paper (refer to [26] for further details on other WSP configurations).

E. Progress to Date

At this stage, we have defined all the 'basic' BN constructs equivalent to the static (AND, OR, K/M), and dynamic (FDEP, PAND, HSP, CSP, WSP, SEQ) gates present in the Galileo dynamic FT. Moreover, we have laid down an algorithm [26] for automatically converting a DFT into a CTBN. We have also defined new constructs, not currently present in the Galileo tool, including mutually exclusive events, deterministic time delay, failure to start on demand, etc. In this manner, we have defined a library of 'basic' BN constructs that possess well-defined semantics. In defining the various BN constructs, we have limited the number of parent nodes (to minimize the BN complexity) to a maximum of 2 for any given node. Nodes, such as the AND gate, having more than two inputs in the DFT, are replaced by multiple gates having two inputs each. The only exception to the above-mentioned rule is when modeling multiple warm spare gates share multiple spares. In this case, some of the nodes have three parent nodes.

The BN model construction and solution is carried out in a modular fashion. The BN constructs defined so far are used to model various system components' behaviors and interactions. More constructs can be readily defined as needed. Defining a new construct consists in finding (1) the right BN structure, and (2) the (conditional) probability distribution for each of the nodes. The 'basic' constructs are combined together to produce more complex constructs, and larger BN (i.e., BN *composition*).

IV. An Application Example

The goal of this section is to show how we can get the closed-form solution of the reliability expression of a given system using the CTBN framework. The procedure is carried out taking advantage of the modular nature of the CTBN framework. The following example is based on the hypothetical cardiac assist system described in [20]. For the sake of illustration, we perform the analysis on the CPU module of this system. Fig. 7 shows the DFT for the CPU module, and the corresponding BN. All components have exponentially distributed failure distributions. P and B have failure rate λ_2, and CS and SS have failure rate λ_1. B's dormancy factor is α. P is the primary of the WSP CPU, and B is its spare unit. The FDEP gate triggers the failure of both P and B. The trigger T is the OR of events CS and SS. CPU_T, in the BN, represents this functional dependency of the CPU unit upon the trigger T. In this case, CPU_T is essentially the OR of the events T and CPU. In fact, instead of having T triggering simultaneously the failure of the primary P, and the spare B (as it is done in the Galileo DFT), we opted, in the BN, for the equivalent scenario where T triggers the failure of the CPU WSP gate as a whole.

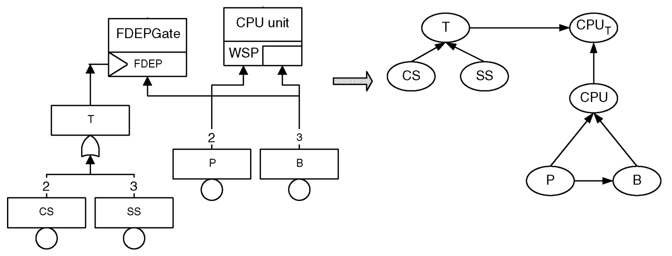

Fig. 7. The CPU module DFT and BN.

The BN on Fig. 7 contains three modules: T, CPU, and CPU_T. We need to find the marginal PDF f_{CPU_T}. We have

$$f_{CPU_T}(t)=\int_0^{\infty}\int_0^{\infty} f_{CPU_T|CPU,T}(t|x,y)f_{CPU}(x)f_T(y)dxdy \tag{17}$$

Let's first find the expressions for f_{CPU}, and for f_T.

$$f_{CPU}(c)=\int_0^{\infty}\int_0^{\infty} f_{CPU|P,B}(c|p,b)f_{B|P}(b|p)f_P(p)dpdb,$$

with (using (4)) $f_{CPU|P,B}(c|p,b) = u(b-p)\delta(c-b)+u(p-b)\delta(c-p)$, and (using (11)) $f_{B|P}(b|p) = u(p-b)\alpha f_{Bi}(b)[1-F_{Bi}(b)]^{\alpha-1}+u(b-p)f_{Bi}(b-p)[1-F_{Bi}(p)]^{\alpha} = u(p-b)\alpha\lambda_2 e^{-\lambda_2 b}e^{-\lambda_2(\alpha-1)b}+u(b-p)\lambda_2 e^{-\lambda_2(b-p)}e^{-\lambda_2\alpha p}$, and $f_P(p)=\lambda_2 e^{-\lambda_2 p}$.

f_{CPU} contains 4 terms; the first term is

$$\begin{aligned}&\int\int_0^{\infty} u(b-p)\delta(c-b)u(p-b)\\&\times\alpha\lambda_2 e^{-\lambda_2 b}e^{-\lambda_2(\alpha-1)b}\lambda_2 e^{-\lambda_2 p}dbdp\\&=\lambda_2^2\alpha\int\int_0^{\infty} u(b-p)\delta(c-b)u(p-b)e^{-\lambda_2\alpha b}e^{-\lambda_2 p}dbdp\\&=\lambda_2^2\alpha e^{-\lambda_2\alpha c}\int_0^{\infty} u(c-p)u(p-c)e^{-\lambda_2 p}dp\end{aligned}$$

The function $f(p)=u(c-p)u(p-c)e^{-\lambda_2 p}$ is zero everywhere except at a single point where $p=c$, and because $f(p)$ is Riemann integrable [27], the integral above, and therefore the first term of f_{CPU}, is equal to zero. The second term of f_{CPU} is

$$\int\int_0^\infty u(b-p)\delta(c-b)u(b-p)\lambda_2 e^{-\lambda_2(b-p)}e^{-\lambda_2\alpha p}\lambda_2 e^{-\lambda_2 p}db dp$$
$$= \lambda_2^2 e^{-\lambda_2 c}\int_0^\infty u(c-p)u(c-p)e^{-\lambda_2\alpha p}dp$$

The function $f(p) = u(c-p)u(c-p)e^{-\lambda_2\alpha p}$ is

$$= \begin{cases} 0 & \text{if } c < p, \\ \frac{1}{4}e^{-\lambda_2\alpha p} & \text{if } c = p, \\ e^{-\lambda_2\alpha p} & \text{if } c > p, \end{cases}$$

Because $f(p)$ is Riemann integrable, changing its value at a single point will not change the value of its integral. In particular, we take, for $p = c$, $f(p) = e^{-\lambda_2\alpha p}$ instead of $f(p) = (1/4)e^{-\lambda_2\alpha p}$. Therefore, the second term of f_{CPU} is equal to

$$\lambda_2^2 e^{-\lambda_2 c}\int_0^\infty u(c-p)u(c-p)e^{-\lambda_2\alpha p}dp$$
$$= \lambda_2^2 e^{-\lambda_2 c}\int_0^c e^{-\lambda_2\alpha p}dp$$
$$= \frac{\lambda_2}{\alpha}e^{-\lambda_2 c}(1-e^{-\lambda_2\alpha c})$$

Using a similar argument made to compute the second term, the third term of f_{CPU} is

$$\int\int_0^\infty u(p-b)\delta(c-p)u(p-b)$$
$$\times\, \alpha\lambda_2 e^{-\lambda_2 b}e^{-\lambda_2(\alpha-1)b}\lambda_2 e^{-\lambda_2 p}db\, dp$$
$$= \lambda_2^2\alpha e^{-\lambda_2 c}\int_0^\infty u(c-b)u(c-b)e^{-\lambda_2\alpha b}db$$
$$= \lambda_2 e^{-\lambda_2 c}(1-e^{-\lambda_2\alpha c})$$

Using a similar argument made to compute the first term, the fourth term of f_{CPU} is

$$\int\int_0^\infty u(p-b)\delta(c-p)u(b-p)$$
$$\times\, \lambda_2 e^{-\lambda_2(b-p)}e^{-\lambda_2\alpha p}\lambda_2 e^{-\lambda_2 p}db\, dp$$
$$= \lambda_2^2 e^{-\lambda_2\alpha c}\int_0^\infty u(c-b)u(b-c)e^{-\lambda_2 b}db$$
$$= 0$$

Thus,

$$f_{CPU}(c) = \frac{\lambda_2}{\alpha}e^{-\lambda_2 c}(1-e^{-\lambda_2\alpha c}) + \lambda_2 e^{-\lambda_2 c}(1-e^{-\lambda_2\alpha c})$$
$$= \lambda_2\frac{1+\alpha}{\alpha}e^{-\lambda_2 c}(1-e^{-\lambda_2\alpha c}) \quad (18)$$

We now compute $f_T(t)$ as

$$f_T(t) = \int_0^\infty\int_0^\infty f_{T|CS,SS}(t|x,y)\, f_{CS}(x)f_{SS}(y)dx\, dy,$$

T is an OR gate, $f_{CS}(x) = \lambda_1 e^{-\lambda_1 x}$, and $f_{SS}(y) = \lambda_1 e^{-\lambda_1 y}$. Thus (see Appendix for the conditional PDF of the OR gate), we have

$$f_T(t) = f_{CS}(t) + f_{SS}(t) - [F_{CS}(t)F_{SS}(t)]'$$
$$= 2\lambda_1 e^{-\lambda_1 t} - \left[(1-e^{-\lambda_1 t})(1-e^{-\lambda_1 t})\right]'$$
$$= 2\lambda_1 e^{-2\lambda_1 t}, \quad (19)$$

and

$$F_T(t) = \int_0^t f_T(x)dx = 1 - e^{-2\lambda_1 t} \quad (20)$$

Now we are able to derive f_{CPU_T}. As noted earlier, CPU_T is essentially an OR gate; therefore

$$f_{CPU_T}(t) = f_{CPU}(t) + f_T(t) - [F_{CPU}(t)F_T(t)]', \quad (21)$$

where

$$f_{CPU}(t) = \lambda_2\frac{1+\alpha}{\alpha}e^{-\lambda_2 t}(1-e^{-\lambda_2\alpha t})$$
$$f_T(t) = 2\lambda_1 e^{-2\lambda_1 t}$$
$$F_{CPU}(t) = \int_0^t f_{CPU}(x)dx$$
$$= \frac{1+\alpha}{\alpha}(1-e^{-\lambda_2 t}) - \frac{1}{\alpha}\left(1-e^{-\lambda_2(1+\alpha)t}\right)$$
$$F_T(t) = \int_0^t f_T(x)dx$$
$$= 1 - e^{-2\lambda_1 t}$$

Therefore, the probability that the CPU module fails during mission time t is

$$Pr(CPU_T \le t) = F_{CPU_T}(t)$$
$$= \int_0^t f_{CPU_T}(x)dx$$
$$= F_{CPU}(t) + F_T(t) - [F_{CPU}(t)F_T(t)]$$
$$= 1 + \frac{e^{-(2\lambda_1+\lambda_2)t}}{\alpha}[e^{-\lambda_2\alpha t} - 1 - \alpha] \quad (22)$$

At this point, having the closed-form analytical solution given in (22), we can perform various analyses and calculations. The CPU module reliability $R(t)$ is

$$R(t) = 1 - F_{CPU_T}(t) = \frac{1+\alpha}{\alpha}e^{-(2\lambda_1+\lambda_2)t} - \frac{1}{\alpha}e^{-(2\lambda_1+\lambda_2+\lambda_2\alpha)t} \quad (23)$$

The mean time to failure (MTTF) is

$$MTTF = \int_0^\infty R(t)dt = \frac{2(\lambda_2+\lambda_1)+\lambda_2\alpha}{(2\lambda_1+\lambda_2+\lambda_2\alpha)(2\lambda_1+\lambda_2)} \quad (24)$$

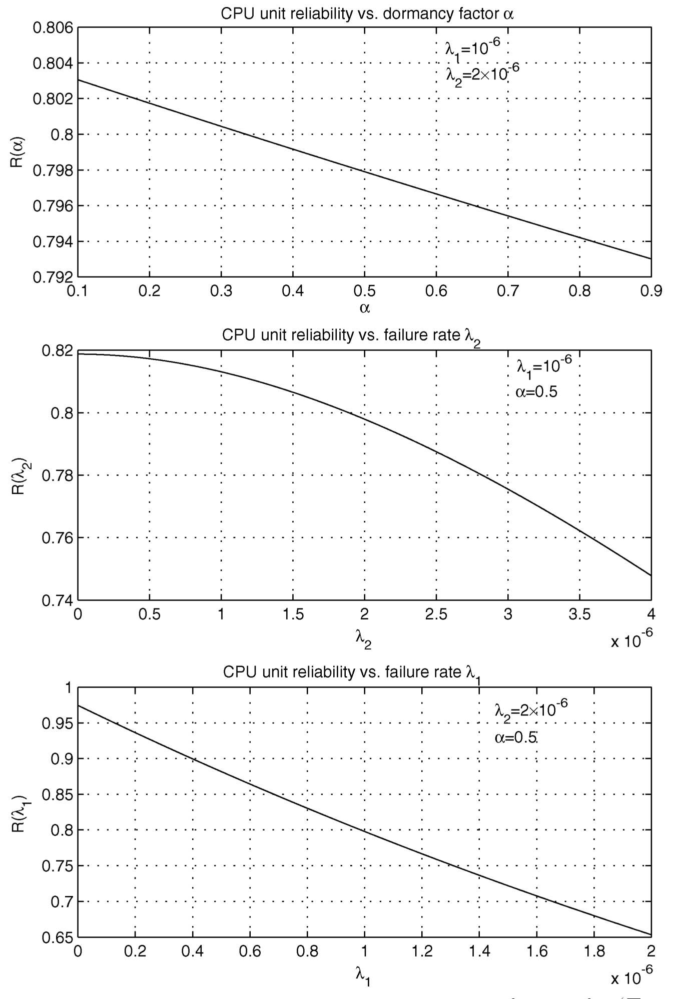

Fig. 8. The CPU module reliability as a function of α, λ_1, and λ_2 ($T = 10^5$ hours).

For a mission time $T = 10^5$ hours (h), $\lambda_1 = 10^{-6}$/h, $\lambda_2 = 2 \times 10^{-6}$/h, and $\alpha = 0.5$. The CPU module reliability is 0.797 899, and the MTTF is 3.5×10^5 hours.

A. Sensitivity Analysis

Fixing the mission time to $T = 10^5$ hours, the CPU module reliability $R(\alpha, \lambda_1, \lambda_2)$ is a function of the three parameters α, λ_1, and λ_2. Fig. 8 shows the variation of R with respect to each of the parameters, keeping the other two parameters constant.

Next, we show the marginal importance, with respect to α, λ_1, and λ_2, of the CPU module reliability by partially differentiating $R(\alpha, \lambda_1, \lambda_2)$. The partial derivative of R with respect to λ_2 is a measure of the marginal importance of components P and B. In addition, the partial derivative of R with respect to α also provides a measure of the marginal importance of B. The partial derivative of R with respect to λ_1 is a measure of the marginal importance of components CS and SS. We have for the various partial derivatives of $R(\alpha, \lambda_1, \lambda_2)$ evaluated at the operating point O (i.e., $T = 10^5$ hours, $\alpha = 0.5$, $\lambda_1 = 10^{-6}$, $\lambda_2 = 2 \times 10^{-6}$)

$$\frac{\partial}{\partial \alpha} R(\alpha, \lambda_1, \lambda_2)\bigg|_O = -\,0.012\,545 \tag{25}$$

$$\frac{\partial}{\partial \lambda_1} R(\alpha, \lambda_1, \lambda_2)\bigg|_O = -\,1.595\,798 \times 10^5 \tag{26}$$

$$\frac{\partial}{\partial \lambda_2} R(\alpha, \lambda_1, \lambda_2)\bigg|_O = -\,0.191\,368 \times 10^5 \tag{27}$$

The negative values in (25), (26), and (27) simply show that the reliability increases as the value of α, λ_1, or λ_2 decreases. Moreover, (26) and (27) suggest that the potential improvement of the CPU module reliability is significantly more important with an improvement in λ_1 than with an improvement in λ_2.

B. Uncertainty Analysis

In this section, we show how to carry out uncertainty analysis on any of the model input parameters (i.e., the components' failure rates, and the dormancy factor in our example). Assume that the failure distribution of SS is exponential with a failure rate β. The failure rate β, however, is not a fixed known value (previously we had the failure rate of SS equal to λ_1), but follows a certain probability distribution f_{B}. This scenario is modeled by adding a new node B, representing the value of the failure rate of component SS, and adding an arc from B to SS; i.e., $\mathrm{B} \rightarrow SS$. Let's assume f_{B} to be uniform in the interval $[\beta_L, \beta_H]$[11]

$$f_{\mathrm{B}}(\beta) = \frac{1}{\beta_H - \beta_L}\left[u(\beta - \beta_L) - u(\beta - \beta_H)\right] \tag{28}$$

Because the failure distribution of SS is exponentially distributed, we write

$$f_{SS|\mathrm{B}}(t|\beta) = \beta e^{-\beta t} \tag{29}$$

Now, we need to find the marginal PDF f_{SS}.[12]

$$\begin{aligned} f_{SS}(t) &= \int_0^\infty f_{SS|\mathrm{B}}(t|\beta) f_{\mathrm{B}}(\beta)\, d\beta \\ &= \frac{1}{\beta_H - \beta_L}\int_{\beta_L}^{\beta_H} \beta e^{-\beta t} d\beta \\ &= \frac{1}{(\beta_H - \beta_L)t}\left[\beta_L e^{-\beta_L t} - \beta_H e^{-\beta_H t}\right] \\ &\quad + \frac{1}{(\beta_H - \beta_L)t^2}\left[e^{-\beta_L t} - e^{-\beta_H t}\right] \end{aligned} \tag{30}$$

The failure distribution F_{SS} is:[13]

$$\begin{aligned} F_{SS}(t) &= \int_0^t f_{SS}(x)\, dx \\ &= \frac{1}{(\beta_H - \beta_L)t}\left[e^{-\beta_H t} - e^{-\beta_L t}\right] \\ &\quad - \lim_{x \to 0} \frac{1}{(\beta_H - \beta_L)x}\left[e^{-\beta_H x} - e^{-\beta_L x}\right], \end{aligned}$$

[11]β_L and β_H are constants.

[12]$\int te^{at}dt = (e^{at}/a)(t - (1/a))$.

[13]$\int (e^{at}/t^2)dt = (-e^{at}/t) + a\int (e^{at}/t)dt$.

and using L'Hôpital's rule,

$$\begin{aligned}
&\lim_{x\to 0}\frac{1}{(\beta_H-\beta_L)x}[e^{-\beta_H x}-e^{-\beta_L x}]\\
&=\frac{1}{(\beta_H-\beta_L)}\lim_{x\to 0}\frac{1}{x}[e^{-\beta_H x}-e^{-\beta_L x}]\\
&=\frac{1}{(\beta_H-\beta_L)}\lim_{x\to 0}[-\beta_H e^{-\beta_H x}+\beta_L e^{-\beta_L x}]\\
&=\frac{1}{(\beta_H-\beta_L)}[-\beta_H+\beta_L]\\
&=-1,
\end{aligned}$$

therefore

$$F_{SS}(t)=\frac{1}{(\beta_H-\beta_L)t}[e^{-\beta_H t}-e^{-\beta_L t}]+1 \qquad (31)$$

Note that f_{SS} is proper. Now, we need to recompute f_T in (19); f_{CS} is unchanged.

$$\begin{aligned}
f_T(t)&=f_{CS}(t)+f_{SS}(t)-[F_{CS}(t)F_{SS}(t)]'\\
&=\frac{1}{(\beta_H-\beta_L)t}e^{-(\lambda_1+\beta_L)t}\left[\lambda_1+\beta_L+\frac{1}{t}\right]\\
&\quad-\frac{1}{(\beta_H-\beta_L)t}e^{-(\lambda_1+\beta_H)t}\left[\lambda_1+\beta_H+\frac{1}{t}\right]
\end{aligned}$$

And similar to the derivation of F_{SS}, the failure distribution F_T is

$$\begin{aligned}
F_T(t)&=\int_0^t f_T(x)dx\\
&=\frac{1}{(\beta_H-\beta_L)t}\left[e^{-(\lambda_1+\beta_H)t}-e^{-(\lambda_1+\beta_L)t}\right]+1
\end{aligned}$$

Finally, using (21), the probability of the CPU module to fail during mission time t is (F_{CPU} is unchanged)

$$\begin{aligned}
Pr(CPU_T\le t)&=F_{CPU_T}(t)\\
&=\int_0^t f_{CPU_T}(x)\,dx\\
&=F_{CPU}(t)+F_T(t)-[F_{CPU}(t)F_T(t)]\\
&=1+\frac{1}{(\beta_H-\beta_L)t}\left[e^{-(\lambda_1+\beta_H)t}-e^{-(\lambda_1+\beta_L)t}\right]\\
&\quad\times\frac{1}{\alpha}\left[(1+\alpha)e^{-\lambda_2 t}-e^{-\lambda_2(1+\alpha)t}\right]
\end{aligned}$$

So far, the states of all the BN variables were defined as the failure times of the corresponding component. In contrasts, we have introduced, in this section, a new node B whose states are the different possible values of a failure rate β. In general, a variable in our BN framework does not necessarily represent a component's failure time; but it can also represent any other quantity or measure (e.g., the value of the failure rate, the dormancy factor, the water level, the temperature reading, etc.) as long as its (conditional) PDF is known.

V. Discussion and Conclusions

The CTBN, described herein, provides a sound framework for system reliability modeling. Similarly to DFT, the CTBN is a modular high-level modeling language, which directly maps physical system components to nodes in the Bayesian graph. The CTBN reliability framework can be used as (1) a standalone framework for reliability modeling/analysis; or (2) integrated into an existing reliability framework (such as the Galileo DFT) to provide enhanced expressive power, and/or an alternative solution technique. The CTBN generalizes the DTBN framework published in earlier work [20]. The DTBN is a discretized CTBN that enables the usage of a standard BN inference algorithm to carry out various analyses.

In a CTBN, the exact expression of all the marginal (i.e., for each of the root nodes) probability distributions must be given. On the contrary, in a DTBN, the marginal probability tables can be filled using raw life data taken from an 'observed failure frequencies (histogram)' database; and this without making any assumption on the types of failure distributions.

The main advantage of a CTBN over a DTBN is that it provides a closed-form solution for the system reliability (or any of its subsystems). However, at this point, in order to get the closed-form solution, one needs to go through a series of symbolic integrations (due to the lack of a theory for exact inference in continuous-time BN with general distributions). This process is, of course, time consuming when carried out by hand, and only feasible for relatively small sized modules. Moreover, a closed-form solution can only be obtained if the integral is analytically solvable. A numerical integration, and therefore an approximate solution, can be performed in the case where the closed-form solution can not be explicitly derived. We are also currently investigating the possibility of automating this process using a 'symbolic integration tool,' such as the one provided in the MATLAB symbolic toolbox.

As mentioned earlier, another advantage of CTBN over DTBN is memory savings, because the (conditional) probabilities are expressed as parametric functions; as opposed to storing and manipulating multi-dimensional tables in DTBN.

A DTBN is a straight forward, easy approximate technique for evaluating a CTBN. *Stochastic simulation* would be another approximate technique for CTBN evaluation. The major advantage of stochastic simulation over the DTBN solution is that there is no need to generate and store probability tables. The investigation of a stochastic simulation solution technique is left for future work. We conjecture that a stochastic simulation solution technique could be more efficient (in terms of accuracy and computation time) than the DTBN solution in evaluating a CTBN. A mixture of truncated exponentials [23], or a mixture of Gaussians [22] might also be used to approximate the various (conditional) PDF of the CTBN model, and use their respective inference theories to solve the CTBN.

Appendix I
Deriving the WSP Conditional PDF

Let's find the conditional PDF of the spare unit B. According to the definition of the WSP gate, the in-isolation hazard rate $h_{Bi}(t)$ of B is reduced by a factor α when B fails before, or at the same time as, A; and the hazard rate remains the same when B fails after A. Assume that $F_{Bi}(b)$ is the in-isolation failure distribution of B; then using $f(t) = (dF(t)/dt) = h(t)e^{-\int_0^t h(\tau)d\tau}$, we have[14]

- For B failing before, or at the same time as $A(b \leq a)$

$$\begin{aligned} f_{B|A,B\leq A}(b|a, b \leq a) &= \alpha h_{Bi}(b) e^{-\int_0^b \alpha h_{Bi}(x)dx} \\ &= \alpha \frac{-r_{Bi}(b)}{R_{Bi}(b)} \left(e^{-\int_0^b h_{Bi}(x)dx} \right)^{\alpha} \\ &= -\alpha r_{Bi}(b) \left[R_{Bi}(b) \right]^{\alpha - 1} \end{aligned} \quad (32)$$

- For B failing after $A(b > a)$

$$\begin{aligned} & f_{B|A,B>A}(b|a, b > a) \\ &= [R_{Bi}(a)]^{\alpha} \frac{-r_{Bi}(b-a)}{R_{Bi}(b-a)} e^{-\int_a^b h_{Bi}(x-a)\, dx} \\ &= [R_{Bi}(a)]^{\alpha} \frac{-r_{Bi}(b-a)}{R_{Bi}(b-a)} e^{-\int_0^{b-a} h_{Bi}(x)dx} \\ &= [R_{Bi}(a)]^{\alpha} \frac{-r_{Bi}(b-a)}{R_{Bi}(b-a)} R_{Bi}(b-a) \\ &= -\left[R_{Bi}(a) \right]^{\alpha} r_{Bi}(b-a) \end{aligned} \quad (33)$$

The factor $[R_{Bi}(a)]^{\alpha}$ is the survival of B until time $t = a$. Indeed, using (32), we have $[R_{Bi}(a)]^{\alpha} = 1 - \int_0^a -\alpha r_{Bi}(b)[R_{Bi}(b)]^{\alpha-1} db$. Concisely, we can write

$$\begin{aligned} f_{B|A}(b|a) = u(a-b) \alpha f_{Bi}(b) \left[1 - F_{Bi}(b)\right]^{\alpha-1} \\ + u(b-a) f_{Bi}(b-a) \left[1 - F_{Bi}(a)\right]^{\alpha} \end{aligned} \quad (34)$$

APPENDIX II
DERIVING THE OR CONDITIONAL PDF

The derivation of the OR gate's output conditional PDF is similar to the AND gate. In fact, the BN structures of the OR and the AND gates are identical. However the OR gate output conditional PDF is

$$f_{X|A,B}(x|a,b) = u(b-a)\delta(x-a) + u(a-b)\delta(x-b) \quad (35)$$

Similar to the AND gate, the OR gate's BN has the following joint probability density function

$$f_{ABX}(a,b,x) = f_{X|A,B}(x|a,b) f_B(b) f_A(a), \quad (36)$$

and the marginal PDF $f_X(x)$ of the gate output X is

$$\begin{aligned} f_X(x) &= \int_0^{\infty}\int_0^{\infty} u(b-a)\delta(x-a) f_B(b) f_A(a) db\, da \\ &\quad + \int_0^{\infty}\int_0^{\infty} u(a-b)\delta(x-b) f_B(b) f_A(a) db\, da \\ &= \int_0^{\infty} f_A(a)\delta(x-a)\left\{1 - F_B(a)\right\} da \\ &\quad + \int_0^{\infty} f_B(b)\delta(x-b)\left\{1 - F_A(b)\right\} db \\ &= f_A(x) + f_B(x) - \left[F_A(x)F_B(x)\right]' \end{aligned} \quad (37)$$

[14]$r(t) = (dR(t)/dt) = -f(t)$, and $R(t) = 1 - F(t)$.

The probability of failure of X in time interval $[0, t]$ is

$$Pr(X \leq t) = \int_0^t f_X(x)dx = F_A(t) + F_B(t) - F_A(t)F_B(t)$$

Note that the above expression can be directly used each time we need to evaluate an OR gate whose inputs are independent. However, if the inputs are dependent in any way, then we need to explicitly recompute $f_X(x)$ as in (37).

REFERENCES

[1] K. J. Sullivan, J. B. Dugan, and D. Coppit, "The Galileo fault tree analysis tool," in *Proceedings of the 29th Annual International Symposium on Fault-Tolerant Computing*, June 1999, pp. 232–235.

[2] Y. Dutuit and A. Rauzy, "A linear time algorithm to find modules of fault trees," *IEEE Transactions on Reliability*, vol. 45, no. 3, pp. 422–425, 1996.

[3] M. A. Boyd, "Dynamic Fault Tree Models: Techniques for Analyses of Advanced Fault Tolerant Computer Systems," PhD dissertation, Dept. of Computer Science, Duke University, 1991.

[4] J. B. Dugan, S. J. Bavuso, and M. A. Boyd, "Fault trees and sequence dependencies," in *Annual Reliability and Maintainability Symposium*, January 1990, pp. 286–293.

[5] ——, "Dynamic fault-tree models for fault-tolerant computer systems," *IEEE Transactions on Reliability*, vol. 41, no. 3, pp. 363–377, September 1992.

[6] E. Charniak, "Bayesian networks without tears," *AI Magazine*, vol. 12, no. 4, pp. 50–63, 1991.

[7] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*: Morgan Kaufmann, 1988.

[8] F. V. Jensen, *An Introduction to Bayesian Networks*: Springer, 1996.

[9] ——, *Bayesian Networks and Decision Graphs*: Springer, 2001.

[10] K. Kanazawa, "Reasoning About Time and Probability," PhD dissertation, Brown University, Providence, Rhode Island, May 1992.

[11] A. E. Nicholson, "Monitoring Discrete Environments Using Dynamic Belief Networks," PhD dissertation, Dept. of Engineering Science, University of Oxford, 1992.

[12] G. Arroyo-Figueroa and L. E. Sucar, "A temporal Bayesian network for diagnosis and prediction," in *Uncertainty in Artificial Intelligence. Proceedings of the 15th UAI Conference*, 1999, pp. 13–20.

[13] S. F. Galán and F. J. Díez, "Modeling dynamic causal interactions with Bayesian networks: temporal noisy gates," in *2nd International Workshop on Causal Networks (CaNew'2000) Held in Conjunction With ECAI 2000*, Berlin, Germany, August 2000, pp. 1–5.

[14] ——, "Networks of probabilistic events in discrete time," *International Journal of Approximate Reasoning*, vol. 30, no. 3, pp. 181–202, September 2002.

[15] U. Nodelman, C. R. Shelton, and D. Koller, "Continuous time Bayesian networks," in *Uncertainty in Artificial Intelligence. Proceedings of the 18th UAI Conference*, 2002.

[16] 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*, vol. 71, no. 3, pp. 249–260, March 2001.

[17] J. G. Torres-Toledano and L. E. Sucar, "Bayesian networks for reliability, analysis of complex systems," *Lecture Notes in Artificial Intelligence*, vol. 1484, pp. 195–206, 1998.

[18] M. Bouissou and O. Pourret, "A Bayesian belief network based method for performance evaluation and troubleshooting of multistate systems," *International Journal of Reliability Quality and Safety Engineering*, vol. 10, no. 4, pp. 407–416, 2003.

[19] P. Weber and L. Jouffe, "Reliability modeling with dynamic Bayesian networks," in *5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS'03)*, Washington D.C., USA, June 2003.

[20] H. Boudali and J. B. Dugan, "A discrete-time Bayesian network reliability modeling and analysis framework," *Reliability Engineering and System Safety*, vol. 87, no. 3, pp. 337–349, March 2005.

[21] ——, "A new Bayesian network approach to solve dynamic fault trees," in *Reliability and Maintainability Symposium*, Jan 2005.

[22] S. L. Lauritzen and F. Jensen, "Stable local computation with conditional Gaussian distributions," *Statistics and Computing*, vol. 11, pp. 191–203, 2001.

[23] S. Moral, R. Rumí, and A. Salmerón, "Mixtures of truncated exponentials in hybrid Bayesian networks," in *Proceedings of the 6th European Conference on Symbolic and Quantitative Approaches to Reasoning With Uncertainty*, vol. 2143, Lecture Notes in Artificial Intelligence, 2001, pp. 145–167.

[24] J. Spanier and K. B. Oldham, *An Atlas of Functions*: Hemisphere publishing corporation, 1987.

[25] R. M. Bracewell, *The Fourier Transform and Its Application (Chapter 5)*, 2nd ed: McGraw-Hill, Inc., 1986.

[26] H. Boudali, "A Bayesian Network Reliability Modeling and Analysis Framework," PhD dissertation, University of Virginia, Charlottesville, VA, May 2005.

[27] H. Kestelman, *Riemann Integration (Chapter 2)*. New York: Dover, 1960.

[28] S. Mahadevan and R. Rebba, "Validation of reliability computational models using Bayes networks," *Reliability Engineering and System Safety*, vol. 87, no. 2, pp. 223–232, February 2005.

[29] C. G. Bai, Q. P. Hu, M. Xie, and S. H. Ng, "Software failure prediction based on a Markov Bayesian network model," *Journal of Systems and Software*, vol. 74, no. 3, pp. 275–282, February 2005.

Hichem Boudali received a Ph.D. degree in Computer Engineering from the School of Engineering and Applied Science at the University of Virginia (Charlottesville, VA, USA), in 2005. He received a Computer Science Engineering degree (M.Sc. equivalent) from the Polytechnic School and Applied Sciences at the Université Libre de Bruxelles (Brussels, Belgium), in 2002. The same year, he obtained a M.Sc. degree in Electrical Engineering from the University of Virginia. He is also a member of IEEE.

Joanne Bechta Dugan was awarded the B.A. degree in Mathematics and Computer Science from La Salle University, Philadelphia, PA in 1980; and the M.Sc., and Ph.D. degrees in Electrical Engineering from Duke University, Durham, NC in 1982, and 1984, respectively. Dr. Dugan is currently a Professor of Electrical and Computer Engineering at the University of Virginia. She has performed and directed research on the development and application of techniques for the analysis of computer systems which are designed to tolerate hardware and software faults. Her research interests thus include hardware and software reliability engineering, fault tolerant computing, and mathematical modeling using dynamic fault trees, Markov models, Petri nets and simulation, and Bayesian networks. Dr. Dugan is an IEEE Fellow, was Associate Editor of the IEEE TRANSACTIONS ON RELIABILITY for 10 years, and is currently Associate Editor of the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. She served on the National Research Council Committee on Application of Digital Instrumentation and Control Systems to Nuclear Power Plant Operations and Safety.