REVIEW OF VARIOUS DYNAMIC MODELING METHODS AND DEVELOPMENT OF AN INTUITIVE MODELING METHOD FOR DYNAMIC SYSTEMS

SEUNG KI SHIN and POONG HYUN SEONG*
Department of Nuclear and Quantum Engineering, KAIST
373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Korea
*Corresponding author. E-mail : phseong@kaist.ac.kr

Received June 26, 2008

Conventional static reliability analysis methods are inadequate for modeling dynamic interactions between components of a system. Various techniques such as dynamic fault tree, dynamic Bayesian networks, and dynamic reliability block diagrams have been proposed for modeling dynamic systems based on improvement of the conventional modeling methods. In this paper, we review these methods briefly and introduce dynamic nodes to the existing reliability graph with general gates (RGGG) as an intuitive modeling method to model dynamic systems. For a quantitative analysis, we use a discrete-time method to convert an RGGG to an equivalent Bayesian network and develop a software tool for generation of probability tables.

KEYWORDS : Dynamic Fault Tree, Reliability Graph with General Gates, Bayesian Network

1. INTRODUCTION

When a new system is developed, an evaluation of the system reliability should be carried out in order to determine whether the system is acceptable for an actual spot. Reliability of systems in a nuclear power plant is also analyzed to compare with regulatory criteria designed to ensure the availability of high-quality qualifications. There are several reliability analysis methods: the fault tree and event tree method, reliability graphs, reliability block diagrams, Markov chains, and Monte Carlo simulations. Each method has its own unique features and those features should be considered when determining the most suitable method.

As digital systems are introduced to nuclear power plants, issues related with reliability analyses of these digital systems are being generated. The conventional static modeling methods, such as the event tree and fault tree methods, present significant shortcomings when used in the reliability modeling of digital instrumentation and control (I&C) systems in nuclear power plants, as they cannot properly account for dynamic interactions between the digital systems and components [1].

To overcome the limitations of the conventional static modeling methods, dynamic modeling methods should be adopted for the reliability analyses of digital systems in nuclear power plants. A dynamic fault tree method based on the introduction of additional dynamic gates to a static fault tree was proposed [2] and has been widely used. However, since it is not a perfect method, other modeling methods for dynamic systems have been proposed based on improvements to existing conventional methods, such as Bayesian networks, Petri nets, reliability block diagrams, and Monte Carlo Simulation. Nevertheless it has been concluded that no single available methodology satisfies all the requirements [1].

In this paper, several dynamic modeling methods are reviewed and a new modeling method that has intuitive modeling power is introduced.

2. STUDIES ON SOME DYNAMIC SYSTEM MODELING METHODOLOGIES

Given the limitations of conventional static modeling methods when applied to dynamic systems, several studies on dynamic modeling methods have been conducted. Most research has involved existing modeling methods in order to integrate easily with static methods, which have already been widely implemented in the system modeling field. Conventional fault tree, Bayesian network, Petri-net, reliability block diagram, and Monte Carlo simulation methods have been upgraded making it possible to analyze the dynamic interactions between components of dynamic systems. In this chapter we review those approaches in both qualitative and quantitative terms.

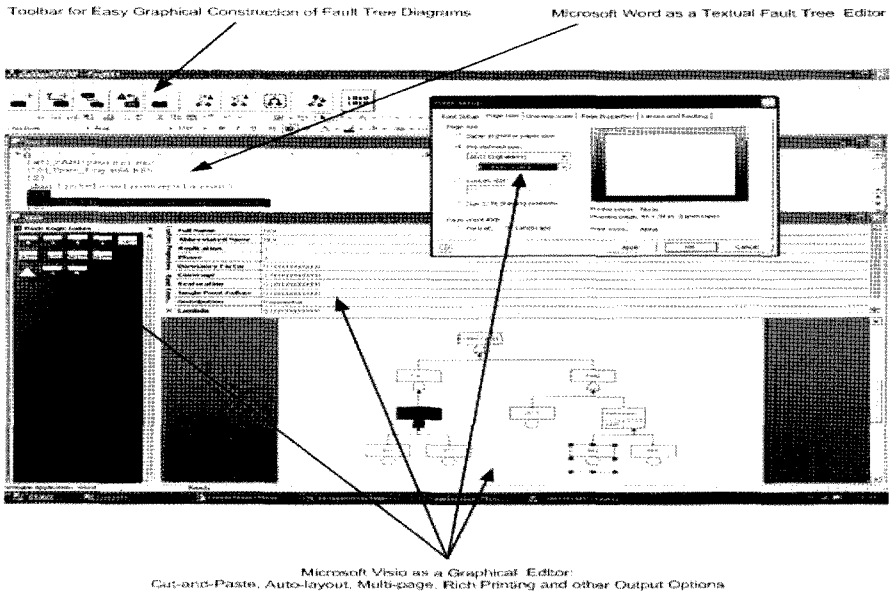

Fig. 1. A Screenshot of Galileo

2.1 Dynamic Fault Tree

A dynamic fault tree (DFT) modeling technique was developed to handle the difficulties that arise in the reliability analysis of fault-tolerant computer systems when critical applications are complicated by several factors [2]. A software tool, Galileo, for DFT modeling and analysis was also developed. Figure 1 shows a screenshot of Galileo [3].

Four dynamic gates were newly adopted in the conventional fault tree method. Each dynamic gate can express each dynamic failure process that is related to the failure sequence of the component parts. Figure 2 shows the four dynamic gates: a functional-dependency (FDEP) gate, a spare gate (a cold spare (CSP) gate, a hot spare (HSP) gate, and a warm spare (WSP) gate), a priority AND gate (PAND), and a sequence-enforcing (SEQ) gate.

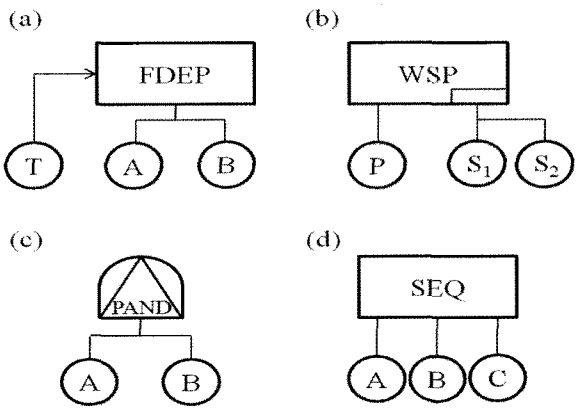

Fig. 2. The Dynamic Gates of a DFT: (a) FDEP; (b) WSP; (c) PAND; and (d) SEQ Gate

2.1.1 Dynamic Gates

2.1.1.1 Functional-Dependency Gate

The FDEP gate has one trigger input and dependent basic events. Whenever a trigger event occurs with this type of gate, all the dependent events are forced to occur.

2.1.1.2 Spare Gate

The spare gate has one primary input and a number of spare inputs. With this gate, the output occurs whenever the primary input and all the spare inputs occur. There are three kinds of spare gates. A CSP gate is used if spare input events never occur in the standby mode. A HSP gate is used if the spare input events have the same probability of occurrence in the standby status as in the active status. A WSP gate with a dormancy factor (α) is used in all other cases.

2.1.1.3 Priority AND Gate

The PAND gate has two inputs. The output of this gate occurs if and only if all input events occur in a particular order. The order of occurrence that causes the output occurrence is from left to right.

2.1.1.4 Sequence-Enforcing Gate

The SEQ gate forces input events to occur in a particular order, namely from left to right, and the output

occurs when all the input events occur. Although the input events of a PAND gate can occur in any order, the input of an SEQ gate cannot occur before the occurrence of an input on the left side.

2.1.2 Quantitative Analysis of DFT

DFTs are solved by conversion to equivalent Markov models [2]. Figure 3 shows Markov chains for the dynamic gates of the DFT [4].

It should be noted that the conversion to a Markov chain generally induces a state space explosion problem as the number of basic events increases [5]. If the number of basic events is n and each event has two conditions, then the total number of states in the Markov chain is 2^n.

The modular approach was proposed to minimize the state space explosion problem [6]. First, a DFT is divided into s-independent sub-trees (modules). If a sub-tree contains a dynamic gate, then it is solved by conversion to a Markov chain and if there is no dynamic gate, then it is solved using Binary Decision Diagrams (BDD); the method to solve static fault trees using BDD can be found in [7, 8], and their solutions are integrated for the calculation of the entire DFT. While the size of the Markov chain can be reduced by this modular approach, modularization techniques are not applied to the solution of a dynamic sub-tree whose top-level gate is a dynamic gate, because it does not provide an exact solution [9].

A Monte Carlo simulation also can be used to

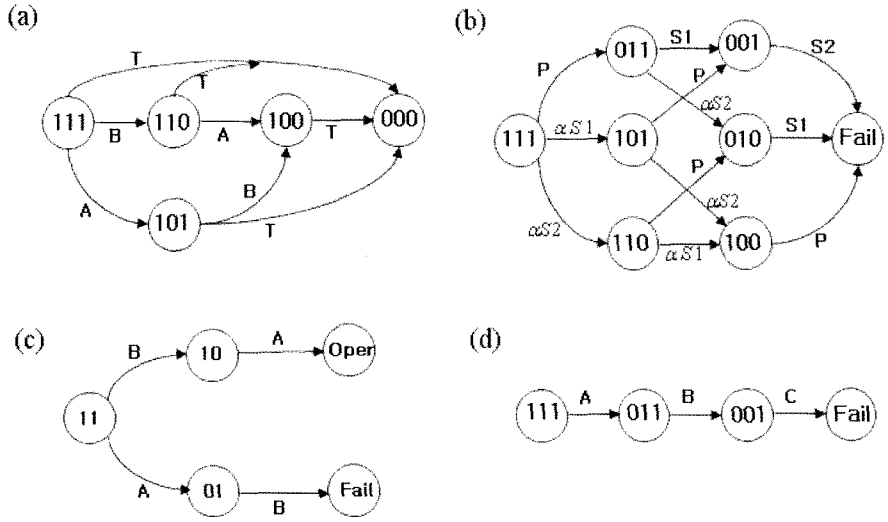

Fig. 3. Markov Chains for the Dynamic Gates: (a) FDEP; (b) WSP; (c) PAND; and (d) SEQ Gate

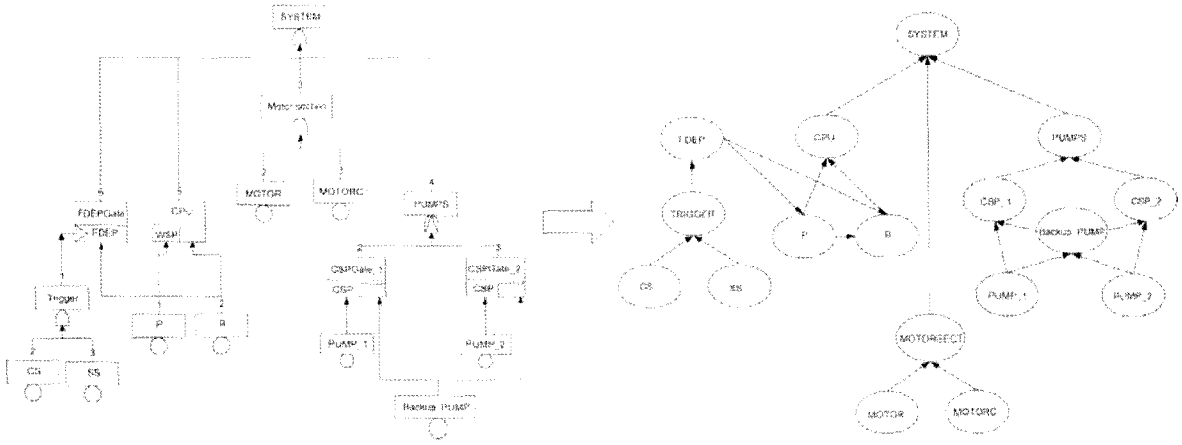

Fig. 4. Conversion of a DFT to Bayesian Network [10]

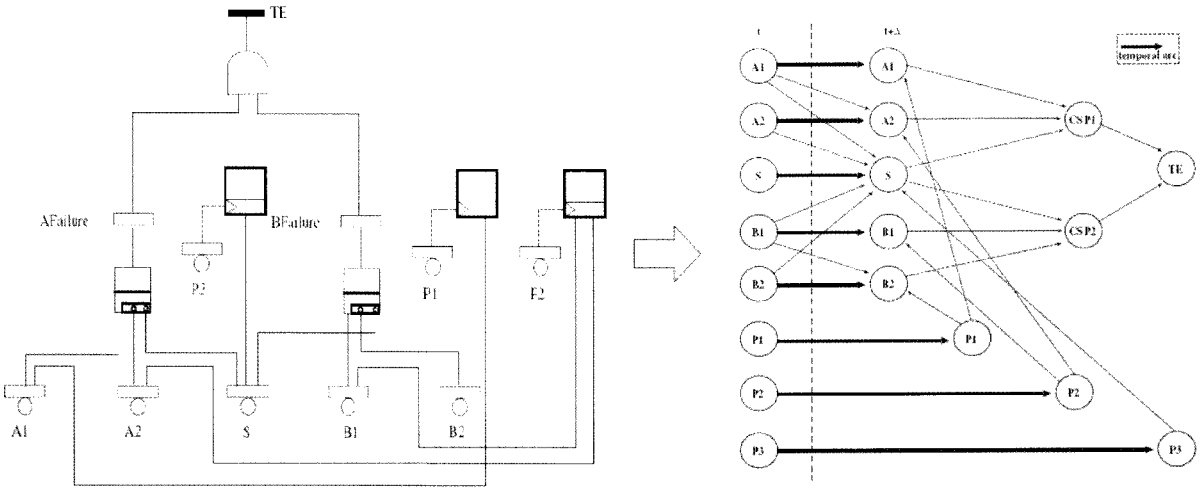

Fig. 5. Conversion of a DFT to Dynamic Bayesian Network [12]

evaluate DFTs without conversion to a Markov chain [9], but generally much computational time is required to obtain results with a desired level of accuracy.

In an attempt to address the limitations of solutions using the Markov model, a solution method using conversion to equivalent Bayesian networks was recently proposed [10-11]. Figure 4 shows an example of the conversion of a DFT to a Bayesian network [10].

The converted Bayesian network can be solved by the discrete-time based method [10] or continuous-time based method [11]. If we use the discrete-time based method, a standard Bayesian network inference algorithm can be employed. However accuracy is unavoidably deficient due to the assumption of discrete-time. The continuous-time based method, meanwhile, provides a closed-form solution for the system reliability, but the exact expression of all the marginal probability distributions must be given and a series of symbolic integrations should be analytically solvable.

The Bayesian network methods noted above transform the DFT to Bayesian network as similar shape. In contrast with those methods, an approach was proposed to convert to a dynamic Bayesian network that has sets of nodes classified by a time axis [12]. In this method, the discrete time assumption is employed and RADYBAN, a tool that allows the user to draw a DFT, automatically converts the DFT into the corresponding dynamic Bayesian network, and asks for reliability measures by means of dynamic Bayesian network inference algorithms.

2.2 Dynamic Modeling Based on a Petri-Net

A Petri-net model is graphically represented by a directed bipartite graph in which the two types of nodes (places and transitions) are drawn as circles, and either bars or boxes, respectively. It can model state transitions using tokens and firing rules. In this section, we review dynamic modeling methods that use Petri-nets.

2.2.1 Conversion of DFTs to Generalized Stochastic Petri nets

Generalized stochastic Petri nets (GSPN) are a performance analysis tool based on a graphical system representation typical of Petri nets, where some transitions are *timed* while others are *immediate* [13] and techniques to generate a continuous time Markov chain (CTMC) from the GSPN and solve the CTMC are available and implemented in several software tools, such as *GreatSPN* [14]. Once the DFT is converted to the equivalent GSPN, the reliability can be obtained using the already available GSPN solver. From this concept, a conversion method of DFTs to GSPN was proposed [15]. The transformation rules for the fault tree events and the static and dynamic gates to the equivalent GSPN were introduced. Figure 6 shows the transformation of the dynamic gates.

However, since this approach also uses the Markov chain, it is hard to avoid the problem raised from the size of states.

To model dynamic performances of systems using Petri nets, a firing delay concept is used in association with transitions. Memory can be introduced to transitions using general distributions for firing delays. Several transitional memory policies were proposed to model the degradation of complex systems [16], but they are insufficient to model the dynamic flexibility as memory is associated with statically positioned transitions. To overcome this shortcoming, memory was assigned to tokens (aging tokens) rather than to transitions [17].

Aging tokens provide dependability modeling flexibility and compact representation of complicated dynamic interactions among multiple entities such as a repairable system with spares, load sharing systems, and systems with shared pools of spares. Figure 7 shows an example of a Petri net with aging tokens.

This method concentrates on descriptive power rather than quantitative analysis power. A tool for this method, named SPN@ was also developed [18] and a Monte Carlo simulation is used to provide quantitative results.

2.3 Dynamic Reliability Block Diagrams

The reliability block diagram (RBD) method is a graphical presentation of a system diagram connecting subsystems of components according to their function or reliability relationships. It is a user-friendly method and it is easy to obtain a model directly from the specifications. From this concept, a new dynamic modeling method, dynamic reliability block diagrams (DRBD), was proposed based on an extension of the extending the existing RBD formalism [19-20].

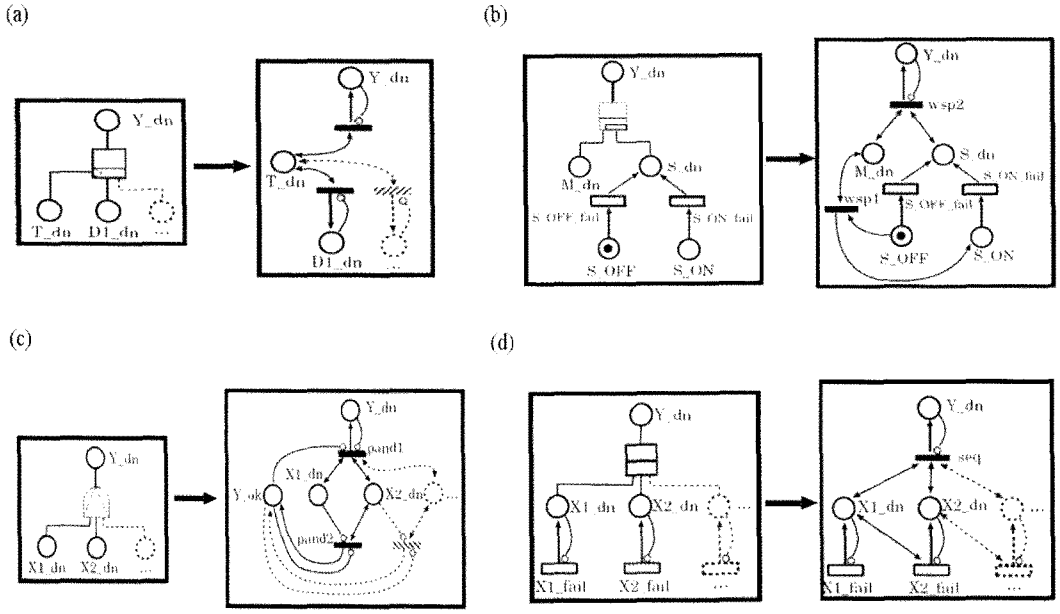

Fig. 6. Transformation Rules for the Dynamic Gates: (a) FDEP; (b) WSP; (c) PAND; and (d) SEQ Gate [15]

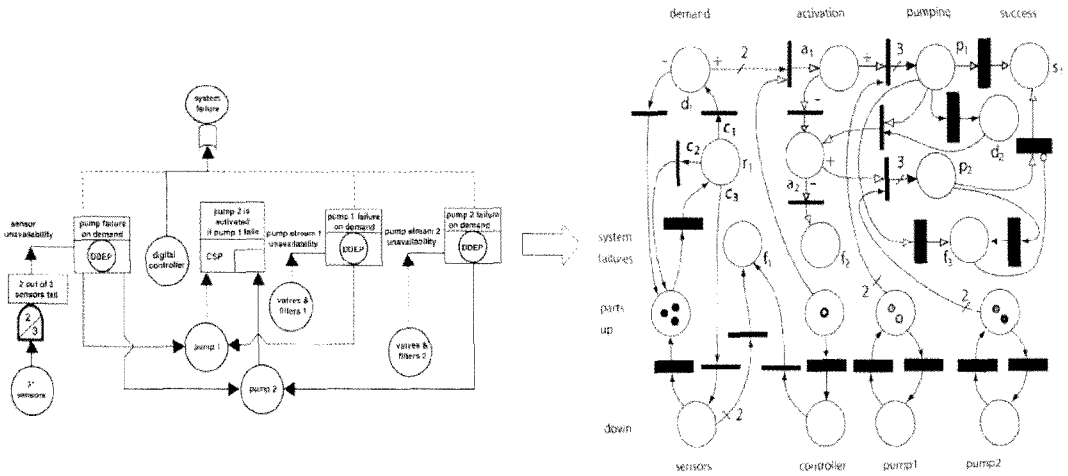

Fig. 7. Models for a Hypothetical Sprinkler System Using DFT and Petri Net with Aging Tokens [17]

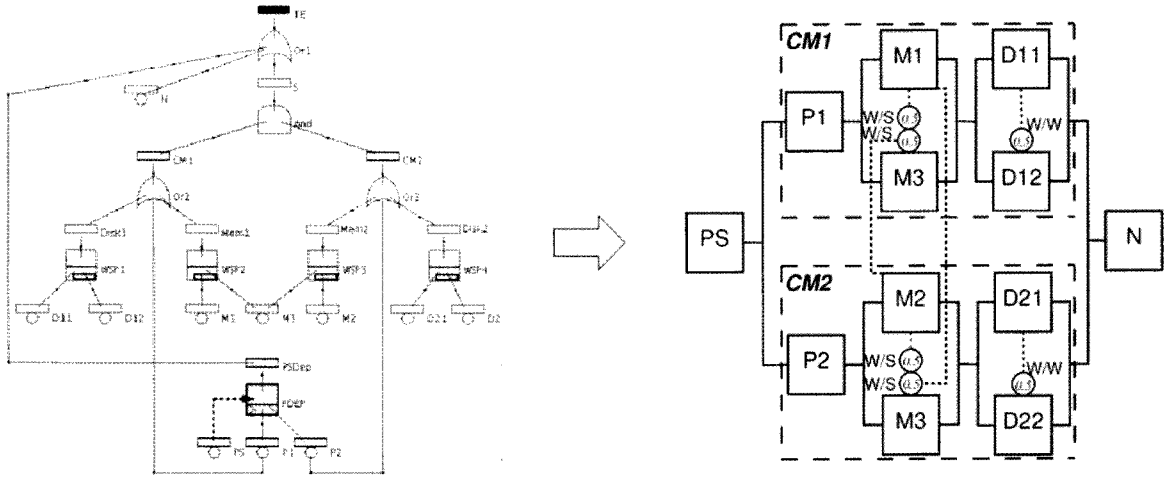

Fig. 8. DFT Model and DRBD Model of a Multiprocessor Distributed Computing System [20]

In a DRBD model, each component is characterized by a variable *state*: *active, failed*, and *standby*. Reliability relations between components are expressed by a specific connection representing the logical condition applied by the *driver* component to the *target* component and the dependency is represented using a dash line with a circle in the target component termination that represents the connection between the related components. 24 different dependencies are characterized and summarized in [19].

The main advantage of DRBD is the capability to model dependencies among subsystems or components concerning their reliability interactions. A quantitative analysis of DRBD can be conducted using the existing methods such as Markov chains and Monte Carlo Simulation.

2.4 Acceleration of Monte Carlo Simulation

Monte Carlo Simulation (MCS) can be used to obtain the reliability from a DFT, but it is time-consuming due to intensive computations. Recently, a method that reduces the calculation time for MCS was proposed based on introduction of the Time-to-Failure tree (TTF-tree) [21]. In this method, both dynamic and static fault trees are converted into TTF-trees using a simple replacement operation, and each TTF-tree is interpreted as a digital circuit, which receives the TTFs of the components and computes the TTF of the whole system. By this procedure, the MCS does not have to repeat identical tasks and the execution time can be reduced.

3. ADDITION OF DYNAMIC NODES TO A RELIABILITY GRAPH WITH GENERAL GATES

Although the fault tree method is the most frequently used method, because it focuses on the causal relations of the failure between components, it cannot intuitively model a system. Consequently, as the system becomes more complex, the fault tree accordingly becomes far more complex. In order to model the system intuitively, a reliability graph with general gates (RGGG) method was proposed; it can make a one-to-one match from the actual structure of a system to the reliability graph of the system [22]. It is based on introduction of general gates to a conventional reliability graph. Therefore, it possesses intuitiveness, which is a characteristic of a conventional reliability graph and additional power of expression.

However, the existing RGGG method is inadequate if the failure of the system is related to the sequence of component failures. Therefore, in this section we improve the existing RGGG method for the purpose of developing an intuitive method that can capture the dynamic behavior of a system failure.

3.1 Reliability Graph with General Gates

A reliability graph is an intuitive reliability analysis method that can model a system by using a one-to-one match graph. However, reliability graphs have not been widely used, because they have a low power of expression; they can only express the characteristics of an OR gate. To overcome the limited power of expression, RGGG was proposed with additional general gates (nodes) [22]. RGGG suffers no loss of intuitiveness and has the advantages of a conventional reliability graph. Figure 9 shows the general nodes (OR, AND, k-out-of-n, and a general purpose node).

Another characteristic of the RGGG is that it has perfect nodes and only the arcs have failure probabilities. As shown in Figure 10, a node and an arc with failure probabilities Pn_j and Pa_j, can be transformed to a perfect

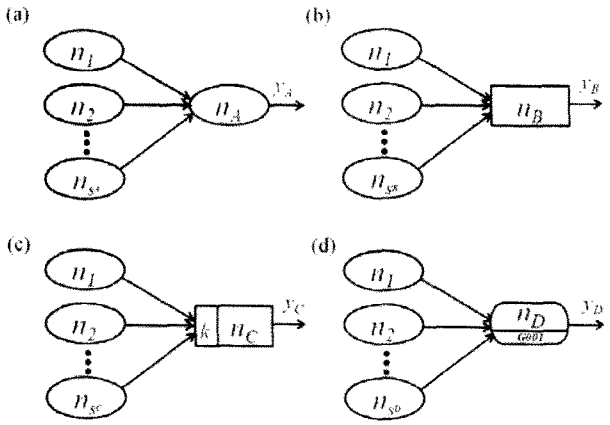

Fig. 9. Definition of the Gates of an RGGG: (a) OR; (b) AND; (c) k-out-of-n; and (d) General Purpose Gate [22]

node and an arc with failure probability of $Pn_j \times Pa_j$.

To calculate the system reliability through the RGGG, RGGG is converted into an equivalent Bayesian network by determining the probability tables of all the nodes. As an example Table 1 shows the probability table of an OR node. A detailed explanation of how to construct a probability table for each node can be found in [22]. In addition to expressing the OR, AND, and k-out-of-n gates, we can express a gate with any characteristic by determining the corresponding probability table.

The intuitive power of the RGGG is confirmed when applied to a real system. As shown in Figure 11, Kim and Seong [22] modeled a digital plant protection system for a nuclear power plant by using an RGGG and a fault tree. A trip case caused by low pressurizer pressure was analyzed. The failure scenario was modeled in only one page with the RGGG but in 64 pages with the fault tree. Furthermore, the RGGG was utilized to readily analyze reliabilities of I&C systems, which consist of sensors and control/protection systems [23].

3.2 Addition of Dynamic Nodes to the RGGG

The RGGG is a good intuitive method but it can only be applied to static systems. To enable the RGGG to model dynamic systems, we introduce additional nodes that can express dynamic processes. In this section, we propose novel dynamic nodes based on the dynamic gates of a dynamic fault tree.

3.2.1 PAND Node

Figure 12(a) shows a PAND node. Node E (n_E) fails only if both signals from node 1 (n_1) and node 2 (n_2) are disconnected and the signal from n_1 is disconnected before the signal from n_2.

3.2.2 Spare Node

Figure 12(b) shows a spare node. The signal from n_1 is a primary input signal and the signal from n_2 is a spare input signal. The letter w indicates that n_F is a WSP node. A spare node fails only if a primary signal and all spare signals are disconnected.

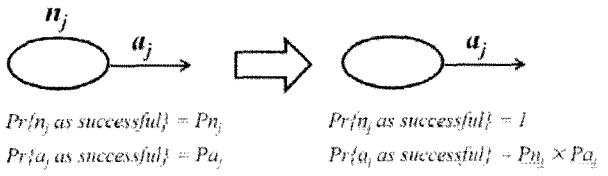

Fig. 10. Transformation to a Reliability Graph with a Perfect Node: (a) the Original Network (with Node and Arc Failures) and (b) the Transformed Network (with Arc Failures Only) [22]

Table 1. A Probability Table for an OR Node with Two Inputs

	y_1=1 (success)		y_1=0 (failure)	
	y_2=1 (success)	y_2=0 (failure)	y_2=1 (success)	y_2=0 (failure)
y_A=1 (success)	$r_{1A}+r_{2A}-r_{1A}r_{2A}$	r_{1A}	r_{2A}	0
y_A=0 (success)	$1-(r_{1A}+r_{2A}-r_{1A}r_{2A})$	$1-r_{1A}$	$1-r_{2A}$	1

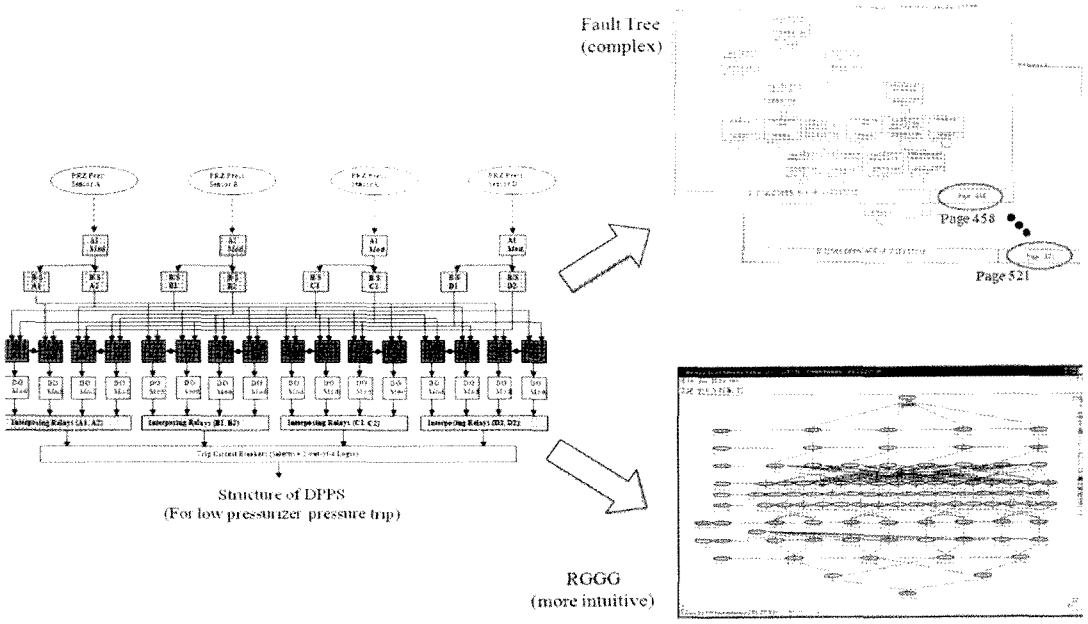

Fig. 11. Modeling of a Digital Plant Protection System with the RGGG and the Fault Tree

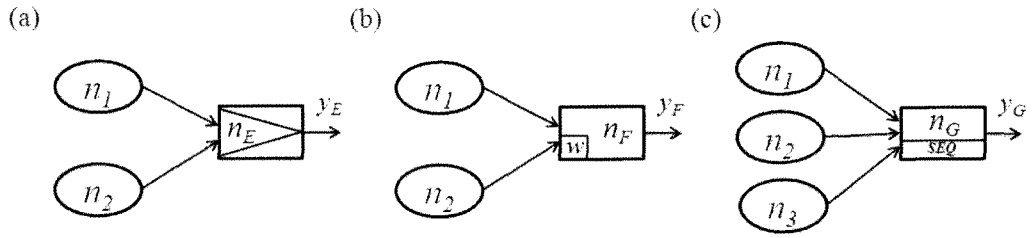

Fig. 12. The Dynamic Nodes of a Dynamic RGGG: (a) PAND; (b) WSP; and (c) SEQ Node

3.2.3 SEQ Node

Figure 12(c) shows an SEQ node. The input signals are constrained to be disconnected in a particular order and the SEQ node fails if and only if all input signals are disconnected. The constrained disconnection order is from top to bottom.

A novel FDEP node is not required in the RGGG. The existing RGGG can illustrate the property of an FDEP gate by using only the OR nodes. As shown in Figure 13, as soon as the signal to n_A is disconnected, none of the signals from n_A can reach the connected nodes n_1, n_2, and n_3. This phenomenon is a property of an FDEP gate of a dynamic fault tree. Thus, no additional node is necessary.

3.3 Quantitative Analysis of Dynamic Nodes

To evaluate an RGGG, we convert the RGGG into an equivalent Bayesian network by determining the probability tables of all the nodes [22]. As novel dynamic nodes are proposed, we also introduce a method to make probability tables for those nodes. We use the discrete-time method to determine the probability tables [10, 24]. In this section, we first briefly describe the discrete-time method and then propose the rules for making the probability table of each node.

3.3.1 The Discrete-Time Method

We divide the line of the total process time (T) into n equal intervals. The time of one interval then becomes t. The output of each node is one of $\{I_1, I_2, \ldots, I_n, I_\infty\}$. If the output of a certain node is I_k, the node fails in the kth interval; and if a node has output I_∞, the node never fails during the total process time.

If P_{ij}^{k} denotes the probability that an arc (a_{ij}) from node i (n_i) to node j (n_j) fails in the kth interval, and if $F_{ij}(t)$ denotes the cumulative failure distribution function

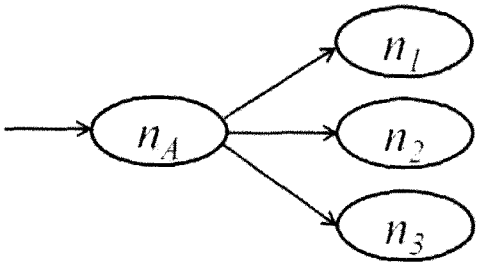

Fig. 13. The Expression of the Property of an FDEP Gate

of a_{ij}, we can derive P_{ij}^k as follows:

$$P_{ij}^k = \int_{(k-1)t}^{kt} \frac{dF_{ij}(t)}{dt} dt \qquad (1)$$

If the total process time and the discretization number (n) are decided, we can derive the probabilities (P_{ij}^k) of all the arcs for all values of k ($k = 1, 2, \ldots, n, \infty$) by using equation (1) before making the probability tables.

3.3.2 Rules of Making the Probability Tables for Dynamic Nodes

To obtain accurate reliability, we should increase the value of n; however, as n increases, the probability table of each node becomes more complex. The number of blanks that should be filled in for a probability table of a node with two inputs is $(n+1)^3$. Figure 14 shows a probability table of a node with two inputs that should be filled in when n is 2. Without rules, the task of making the tables is difficult and prone to errors. We therefore propose a set of rules to establish the probability table of each dynamic node. First, let the outputs of n_1, n_2, n_3, n_E, n_F, and n_G be I_x, I_y, I_z, I_e, I_f, and I_g, respectively ($x, y, z, e, f, g \in 1, 2, \ldots, n, \infty$). Note also that the nodes are perfect nodes in the RGGG method.

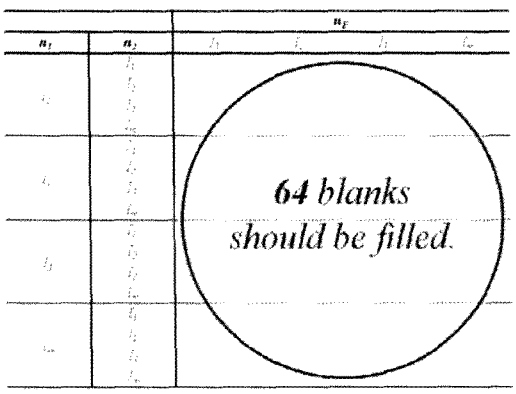

Fig. 14. A Probability Table that should be Filled in

3.3.2.1 PAND Node

We explain the rules for making the probability table of the PAND node shown in Figure 12(a). In the table, each blank, which is defined by x, y, and e, can be filled on the basis of the following rules:

i. If $e > y$,
   0.
ii. If $e = y \leq x$,
   $Pr\{a_{1E}$ fails before the eth interval$\} \cdot (1 - Pr\{a_{2E}$ fails before the eth interval$\})$.
iii. If $e \leq x$, $e < y$,
   $Pr\{a_{1E}$ fails before the eth interval$\} \cdot Pr\{a_{2E}$ fails at the eth interval$\}$.
iv. If $e = y > x$,
   $1 - Pr\{a_{2E}$ fails before the eth interval$\}$.
v. If $x < e < y$,
   $Pr\{a_{2E}$ fails at the eth interval$\}$.
vi. If $e = \infty$,
   1 – (sum of the other probabilities in the same row).

When we fill in the probability table by using rules 1 to 5, we exclude the case where the value of e is ∞. Furthermore, because the sum of the probabilities in each row should be 1, we apply rule 6.

3.3.2.2 Spare Node

We explain the rules for making the probability table of the WSP node shown in Figure 12(b). Because the CSP and HSP nodes are types of WSP nodes (where the dormancy factor (α) is 0 for CSP and 1 for HSP), we describe only the WSP node. In the table, each blank, which is defined by x, y, and f, can be filled on the basis of the following rules:

i. If $f > x, y$,
   0.
ii. If $f < x, y$,
   $Pr\{a_{1F}$ fails at the fth interval$\} \cdot Pr\{a_{2F}$ fails at or before the fth interval$\} + Pr\{a_{1F}$ fails before the fth interval$\} \cdot P\{a_{2F}$ fails at the fth interval$\}$.
iii. If $x < f < y$,
   $Pr\{a_{2F}$ fails at the fth interval$\}$.
iv. If $y \leq f < x$,
   $Pr\{a_{1F}$ fails at the fth interval$\}$.
v. If $f = x < y$,
   $Pr\{a_{1F}$ does not fail before the fth interval$\} \cdot Pr\{a_{2F}$ fails at or before the fth interval$\} + Pr\{a_{1F}$ fails before the fth interval$\} \cdot Pr\{a_{2F}$ fails at the fth interval$\}$.
vi. Else,
   1 – (sum of the other probabilities in the same row).

When we calculate the value of $Pr\{a_{2F}$ fails at the fth interval$\}$ in rules 2, 3, and 5, we must distinguish the period of the spare status, a_{2F}, from the period of the active status, a_{2F}, because the failure rates of each status differ in terms of the dormancy factor. In the dynamic fault tree, the inputs of the spare gate are only basic events [2]. If the RGGG also allows a spare node to have only basic events, which means that n_1 and n_2 have no input, the probability table can be filled on the basis of rules 2 and 6, because x and y are both set as ∞. The task of filling in the table therefore becomes simple.

3.3.2.3 SEQ Node

We explain the rules for making the probability table of the SEQ node shown in Figure 12(c). The SEQ node only allows basic events as inputs except n_1, because if n_2 and n_3 have inputs, the SEQ node cannot constrain the failure order of the inputs. We therefore explain only one case in which y and z are ∞. Each blank under this case can be filled on the basis of the following rules:

i. If $g < 3$,
   0.
ii. If $3 \le g < x+2$,
   $\sum_{\forall a,b,c} Pr\{a_{1G}$ fails at the ath interval$\} \cdot Pr\{a_{2G}$ fails at the bth interval$\} \cdot Pr\{a_{3G}$ fails at the cth interval$\}$, when $a + b + c = g$.
iii. If $g \ge x+2$,
   $Pr_1 + Pr_2$.
   $Pr_1 = \sum_{\forall a,b,c} Pr\{a_{1G}$ fails at the ath interval$\} \cdot Pr\{a_{2G}$ fails at the bth interval$\} \cdot Pr\{a_{3G}$ fails at the cth interval$\}$, when $1 \le a < x-1$, and $a + b + c = g$.
   $Pr_2 = Pr\{a_{1G}$ doesn't fail before the xth interval$\} \cdot \sum_{\forall b,c} Pr\{a_{2G}$ fails at the bth interval$\} \cdot Pr\{a_{3G}$ fails at the cth interval$\}$, when $b + c = g - x$.
iv. If $g = \infty$,
   1 – (sum of the other probabilities in the same row).

In rule 2, the bth interval does not mean the real bth interval in the total process time; rather, it means that a_{2G} fails after b intervals from the interval in which a_{1G} fails. For example, if $g = 7$ and $a = 2$, $b = 3$, and $c = 2$, then a_{1G} fails at the 2nd interval, a_{2G} fails at the 5th interval, and a_{3G} fails at the 7th interval. This outcome is due to the properties of the SEQ node, where a_{2G} cannot fail before a_{1G} fails and a_{3G} cannot fail before a_{2G} fails.

Table 2 shows the probability table for a PAND node; the table is based on the rules of the PAND node for $n = 3$.

Table 2. A Probability Table for a PAND Node when $n=3$

		n_E			
n_1	n_2	I_1	I_2	I_3	I_∞
I_1	I_1	0	0	0	1
	I_2	0	$1-P_{2E}^1$	0	$1-\Sigma$
	I_3	0	P_{2E}^2	$1-P_{2E}^1-P_{2E}^2$	$1-\Sigma$
	I_∞	0	P_{2E}^2	P_{2E}^3	$1-\Sigma$
I_2	I_1	0	0	0	1
	I_2	0	$P_{1E}^1(1-P_{2E}^1)$	0	$1-\Sigma$
	I_3	0	$P_{1E}^1 P_{2E}^2$	$1-P_{2E}^1-P_{2E}^2$	$1-\Sigma$
	I_∞	0	$P_{1E}^1 P_{2E}^2$	P_{2E}^3	$1-\Sigma$
I_3	I_1	0	0	0	1
	I_2	0	$P_{1E}^1(1-P_{2E}^1)$	0	$1-\Sigma$
	I_3	0	$P_{1E}^1 P_{2E}^2$	$(P_{1E}^1+P_{1E}^2)(1-P_{2E}^1-P_{2E}^2)$	$1-\Sigma$
	I_∞	0	$P_{1E}^1 P_{2E}^2$	$(P_{1E}^1+P_{1E}^2)P_{2E}^3$	$1-\Sigma$
I_∞	I_1	0	0	0	1
	I_2	0	$P_{1E}^1(1-P_{2E}^1)$	0	$1-\Sigma$
	I_3	0	$P_{1E}^1 P_{2E}^2$	$(P_{1E}^1+P_{1E}^2)(1-P_{2E}^1-P_{2E}^2)$	$1-\Sigma$
	I_∞	0	$P_{1E}^1 P_{2E}^2$	$(P_{1E}^1+P_{1E}^2)P_{2E}^3$	$1-\Sigma$

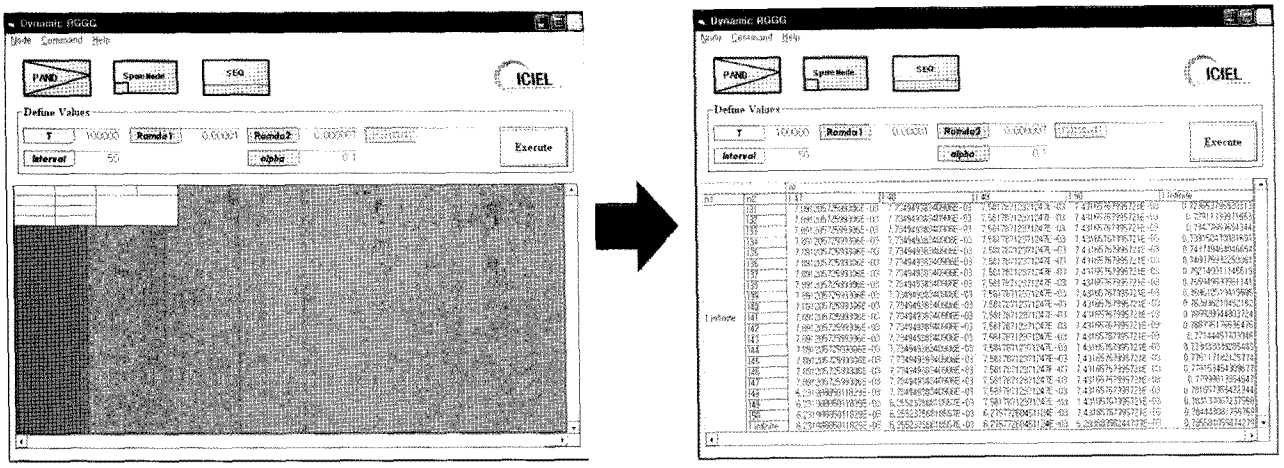

Fig. 15. A Screenshot of the Developed Tool

3.4 Development of a Software Tool

As we use a discrete-time concept, a large number of n is required to obtain an accurate result. However, this leads to complexity in making probability tables. Even though we have established rules for making probability tables, it is not easy to make the tables without help from a software tool. In this section, we briefly introduce a software tool that generates probability tables automatically. Figure 15 shows a screenshot of the developed tool. First, we select one node among three dynamic nodes and input all the variables. Then, an appropriate probability table is generated automatically as shown in the right side of Figure 15. We found that the calculation results become more accurate as n increases.

To analyze a dynamic system, it is necessary to connect and calculate the probability tables of all the nodes in the model. Thus far, we have developed the tool only for the generation of the probability table of each node. We are currently developing a tool to connect the generated probability tables with an existing Bayesian software tool, such as Microsoft Belief Networks or BNet.

4. SUMMARY

There are many reliability analysis methods, such as fault trees, reliability graphs, reliability block diagrams, Markov chains, and Monte Carlo simulations. However, most of these methods analyze the static systems of the failure process rather than the sequence of component failures. Various techniques have been proposed to model dynamic systems on improvements of conventional modeling methods. In this paper, we briefly reviewed those methods. As it is difficult to select an optimal method that is suitable to every dynamic system, it is necessary to choose the technique that is most adequate for the target system.

In the present paper, we proposed an intuitive modeling method for dynamic systems. We upgraded the capability of the RGGG so that it can be used to model dynamic systems. First, we introduced novel dynamic nodes to the RGGG on the basis of the dynamic gates of a dynamic fault tree. Next, we developed a method to evaluate the new dynamic nodes. To evaluate the RGGG, we transformed it to an equivalent Bayesian network by determining a probability table for each node. Using a discrete-time method to obtain the probability tables of the dynamic nodes, we found that the tables become large and more difficult to establish. We consequently proposed a set of rules for making the probability tables of the dynamic nodes to reduce time and errors. Once the RGGG is converted to an equivalent Bayesian network, the system reliability can be easily obtained with various commercial or free software tools, such as Microsoft Belief Networks. A software tool for generation of probability tables was developed on the basis of the proposed rules and we are currently developing a tool to connect the generated probability tables with an existing Bayesian software tool.

ACKNOWLEDGEMENTS

This work was supported by the Acceleration Research Program.

REFERENCES

[1] T. Aldemir, D.W. Miller, M. Stovsky, J. Kirschenbaum, P. Bucci, A.W. Fentiman, and L.T. Mangan, "Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments," NUREG/CR-6901, U.S. NRC (2006).

[2] J.B. Dugan, S.J. Bavuso, and M.A. Boyd, "Dynamic Fault -Tree Models for Fault-Tolerant Computer Systems," IEEE Transactions on Reliability, **41**, 363 (1992).

[3] K.J. Sullivan, J.B. Dugan, and D. Coppit, "The Galileo Fault Tree Analysis Tool," Proceedings of Annual International Symposium on Fault-Tolerant computing, Madison, USA, Jun. 15-18, 1999.
[4] R. Manian, D.W. Coppit, K.J. Sullivan, and J.B. Dugan, "Bridging the Gap between Systems and Dynamic Fault Tree Models," Proceedings of Annual Reliability and Maintainability Symposium, Washington DC, USA, Jan. 18-21, 1999.
[5] S. Amari, G. Dill, and E. Howald, "A New Approach to Solve Dynamic Fault Trees," Proceedings of Annual Reliability and Maintainability Symposium, Tampa, USA, Jan. 27-30, 2003.
[6] R. Gulati and J.B. Dugan, "A Modular Approach for Analyzing Static and Dynamic Fault Trees," Proceedings of Annual Reliability and Maintainability Symposium, Philadelphia, USA, Jan. 13-16, 1997
[7] R.M. Sinnamon and J.D. Andrews, "Quantitative Fault Tree Analysis using Binary Decision Diagrams," European Journal of Automation, **30**, 1051 (1996).
[8] A. Rauzy, "New Algorithms for Fault Tree Analysis," Reliability Engineering and System Safety, **40**, 203 (1993).
[9] J.B. 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**, 49 (2000).
[10] H. Boudali and J.B. Dugan, "A Discrete-time Bayesian Network Reliability Modeling and Analysis Framework," Reliability Engineering and System Safety, **87**, 337 (2005).
[11] H. Boudali and J.B. Dugan, "A Contiuous-Time Bayesian Network Reliability Modeling, and Analysis Framework," IEEE Transactions of Reliability, **55**, 86 (2006).
[12] S. Montani, L. Portinale, A. Bobbio, D. Codetta-Raiteri, "Radyban: A Tool for Reliability Analysis of Dynamic Fault Trees through Conversion into Dynamic Bayesian Networks," Reliability Engineering & System Safety, **93**, 922 (2008).
[13] G. Chiola, M.A. Marsan, G. Balbo, and G. Conte, "Generalized Stochastic Petri Nets: A Definition at the Net Level and its Implications," IEEE Transactions on Software Engineering, **19**, 89 (1993).
[14] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo, "GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets," Performance Evaluation, **24**, 47 (1995).
[15] D. Codetta-Raiteri, "The Conversion of Dynamic Fault Trees to Stochastic Petri Nets, as a Case of Graph Transformation," Electronic Notes in Theoretical Computer Science, **127**, 45 (2005).
[16] A. Bobbio, A. Puliafito, and M. Tekel, "A Modeling Framework to Implement Preemption Policies in Non-Markovian SPNs," IEEE Transactions on Software Engineering, **26**, 36 (2000).
[17] V. Volovoi, "Modeling of System Reliability using Petri Nets with Aging Tokens," Reliability Engineering & System Safety, **84**, 149 (2004).
[18] V. Volovoi, "Stochastic Petri Nets Modeling using SPN@," Proceedings of Annual Reliability and Maintainability Symposium, Newport Beach, USA, Jan. 23-26, 2006.
[19] S. Distefano, L. Xing, "A New Approach to Modeling the System Reliability: Dynamic Reliability Block Diagrams," Proceedings of Annual Reliability and Maintainability Symposium, Newport Beach, USA, Jan. 23-26, 2006.
[20] S. Distefano, A. Puliafito, "Dependability Modeling and Analysis in Dynamic Systems," Proceedings of Parallel and Distributed Processing Symposium, Long Beach, USA, Mar. 26-30, 2007.
[21] A. Ejlali, S.G. Miremadi, "FPGA-based Monte Carlo Simulation for Fault Tree Analysis," Microelectronics Reliability, **44**, 1017 (2004).
[22] M.C. Kim and P.H. Seong, "Reliability Graph with General Gates: an Intuitive and Practical Method for System Reliability Analysis," Reliability Engineering and System Safety, **78**, 239 (2002).
[23] M.C. Kim and P.H. Seong, "A computational method for probabilistic safety assessment of I&C systems and human operators in nuclear power plants," Reliability Engineering and System Safety, **91**, 580 (2006).
[24] S.F. Galán, F.J. Diez, "Networks of Probabilistic Events in Discrete Time," International Journal of Approximate Reasoning, **30**, 181 (2002).