Parametric Dynamic Fault Tree and its Solution through Modularization

Daniele Codetta Raiteri
Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino, Italy
codetta@di.unito.it

1. Introduction

The *Fault Tree* (FT) [1] is a widespread model for the dependability analysis of complex systems and allows to model how combinations of component failures can determine the failure of subsystems or of the whole system; an example is reported in Fig. 1; the nodes can be failure events or gates: failure events are represented as a bar and are equivalent to a boolean variable whose value is 0 until the failure event has not yet occured, or 1 when the event has occured; gates are connected to events by means of arcs and have several input events and a unique output event, connected respectively below and above the gate. The events that are represented as a bar with a circle, are called *basic events* (BE) and correspond to the failure events of physical components of the system; the occurence time of such events is ruled by a probability distribution associated to each BE, typically an exponential distribution whose parameter λ is called *failure rate*. The internal events, represented as an empty bar, correspond to subsystems failures; an internal event is the output of a gate and occurs when a particular combination (determined by the type of the gate) of the gate input events occurs. While BEs can not be the output of any gate, there is a unique event called *Top Event* (TE), indicated by a black bar, that can only be the output of a gate; TE represents the failure of the whole system. A FT may contain several kinds of gates; in the standard version of this model, BEs are considered as statistically indipendent and three gates corresponding to the *AND, OR, K of N* boolean functions are defined; such a model can be easily analyzed in a combinatorial way [1], but it suffers from the inability to represent dependencies among failure events; in order to overcome this limitation, some new gates called *dynamic gates* were introduced and they are:

- *Priority AND (PAND)* - it fails if all of its input events fail and in a specified order (from left to right);
- *Functional Dependency Gate (FDEP)* - it forces a set of dependent events to occur when a particular event called *Trigger* occurs;
- *Sequence Enforcing Gate (SEQ)* - it forces a set of events to occur in a specified order (from left to right);
- *Warm Spare Gate (WSP)* - it models the presence of a set of spare components able to replace one or more main components when one of them fails; the spares change their failure rate when turning from the dormant state (stand-by) to the working state.

The resulting model is called *Dynamic Fault Tree* (DFT) [2]. The introduction of dynamic gates changes the way to perform the system dependability analysis: while standard FTs are solved in a combinatorial way, DFTs require the state space analysis. Another evolution of the model concerns the way to represent the system redundancies and symmetries compactly; this purpose is achieved by the *Parametric Fault Tree* (PFT) [3]; using PFT, identical subsystems are represented by a unique parameterized subtree whose root is a *(Basic) Replicator Event* (RE), indicated by a dotted bar; a parameter is associated to the RE and its variation range (for instance, from 1 to 3) indicates how many identical subtrees are represented in a compact way; such parameter will be associated even to the events inside the replicated subtree; a replicated subtree may furtherly contain other REs using combinations of several parameters. DFT and PFT formalisms can be integrated generating the *Parametric Dynamic Fault Tree* (PDFT) [4] supporting both parameterization and dependencies in the failure mode. As DFT, PDFT needs the state space solution.

2. The solution technique

Fig. 1 is an example of DPFT: this system is composed by three subsystems called $SYS1$, $SYS2$, $SYS3$, and it fails when two of them are failed (TE is the output of a K of N gate with $k = 2$ and $n = 3$); $SYS1$ (output of an OR gate) fails if at least one among $SUB1$, $SUB2$ and $SUB3$ (represented as $SUB(i)$ with i varying from 1 to 3) fails; $SUB(i)$ fails when all of its components ($A(i)$ and $B(i,j)$ with j varying from 1 to 2) fail ($SUB(i)$ is the output of an AND gate). $SYS2$ is the output of a PAND gate, so it fails if both C and D_F are failed and C failed before D_F; D_F

fails if at least one $D(k)$ is failed; $SYS3$ fails if E or M_F fails; M_F (output of a WSP gate) fails when the main component M is failed and there are no spares $SP(h)$ available to replace it.

As mentioned in section 1, PDFT requires the state space solution because it can contain dependencies in the failure mode due to dynamic gates; state space solution may be computationally very expensive, so in our approach we try to apply such technique only to those subtrees that specifically require it. So, the first step of the adopted solution process is called *modularization* and consists of detecting modules, i. e. indipendent subtrees; a subtree is a module if both it does not share any node with other subtrees and it does not descend from a dynamic gate; modules are detected applying a previously realized linear time algorithm [5] that has been adapted to consider the presence of dependencies due to dynamic gates [6] or shared parameterized nodes [4]. Once modules are detected, they are classified as dynamic or static if they contain or not at least one dynamic gate (*classification* step). Dynamic modules are then classified as *minimal* if they do not contain other modules of any kind.

Only dynamic modules need a state space solution; a way to perform such analysis consists of translating them in a High Level Stochastic Petri Net in the form of *Stochastic Well-formed Net* (SWN) [7]; this is done by means of a specific translator (Fig. 3, Fig. 4). Suppose we have to perform a transient analysis of the PDFT, in other words we have to calculate the probability that the system is failed at a given mission time: we detach each minimal dynamic module (MDM) and we translate it in a SWN. Instead of the ordinary state space, a symbolic state space, whose dimensions are smaller, can be generated from a SWN and analized; in this way, we solve in isolation every MDM, calculating its probability of failure at the given mission time (*decomposition* step). The combined use of parameterization, modularization, decomposition and translation of MDMs in SWN can lead to a relevant reduction of the state space dimensions [4].

The translation of a MDM in SWN is performed in this way: for each BE and for each gate inside the MDM, a SWN is created separately; considering the MDM in Fig. 1 whose root is $SYS2$, we create a distinct SWN for the BEs C (Fig. 2.a) and $D(k)$ (Fig. 2.b), and for the gates PAND (Fig. 2.c) and OR (Fig. 2.d); then, such SWNs are composed together performing a superposition over the common places corresponding to MDM events; in this case, C, $D(k)$ and D_F. The resulting net is shown in Fig. 3; SWN compositionality is very flexible allowing the generation of the corresponding SWN even for complicated combinations of gates.

Now we replace in the PDFT each detached MDM with a BE which has not a failure rate but a probability of failure that is equal to the probability calculated on the corresponding module (*substitution* step); at this point we obtain a PFT that is no more dynamic because it does not contain any dynamic gate; we call it *Reduced PFT* (Fig. 5) and it can be solved in a combinatorial way after having been unfolded, i. e. converted in the equivalent FT.

The whole process has been implemented following a multi-solution multi-formalism approach [8] adapting the graphical tool called *DrawNET++* [9] to the PDFT formalism, using the *GreatSPN* tool [10] as SWN solver and the *SHARPE* package [11] to solve the reduced PFT, once unfolded.

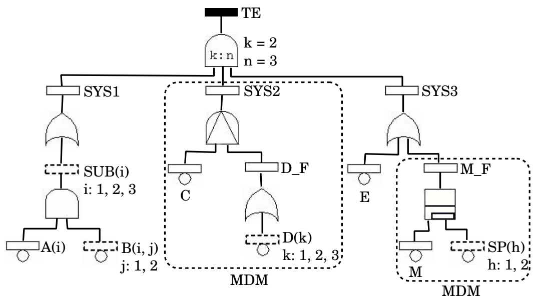

Figure 1. An example of DPFT

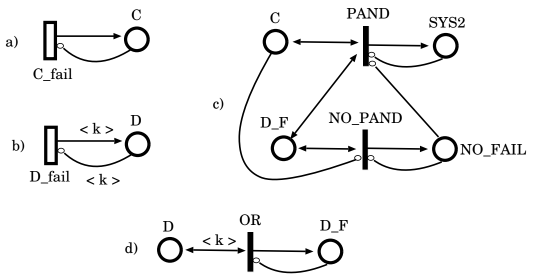

Figure 2. SWNs for each BE and gate in SYS2

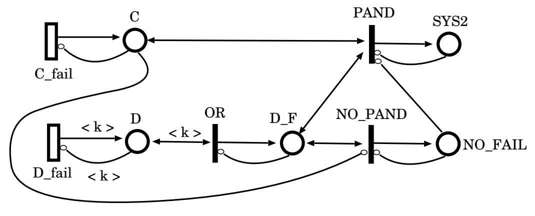

Figure 3. SWN for the module SYS2

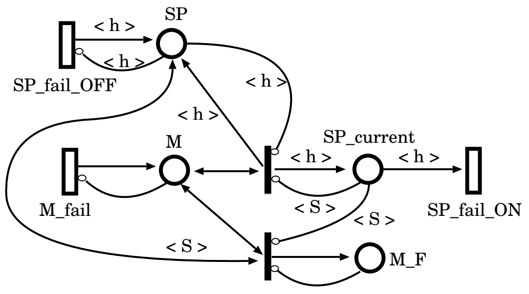

Figure 4. SWN for the module M_F

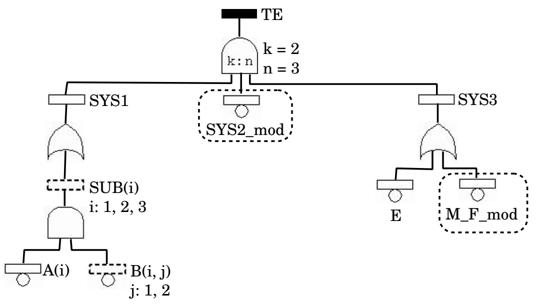

Figure 5. The reduced PFT

3. Future works

Future developing on PDFT will regard mainly these aspects:

- the integration of the PDFT formalism with the Repairable Fault Tree (RFT) [12] formalism; this issue has alreay been partially studied in [4], but only regarding the WSP gate; failure and repair semantics must be defined in order to integrate dynamic gates and *Repair Boxes* (RB) [4][12]; for instance, if one of the input events of a PAND gate is repairable, such event is repeatable: which is the failure order causing the failure of the PAND gate?
- Currently only the quantitative analysis is available for PDFT, so a way to perform a qualitative analysis of PDFT must be studied in order to detect minimal cut sets (MCS) or sequences, i. e. the minimal sets or sequences of BEs leading to the failure of the whole system; this issue was already partially faced in the case of PFT [3] and DFT [13].
- The solution of the reduced PFT requires an unfolding step; we would like to be able to perform a combinatorial analysis of a PFT directly, without unfolding it; this might be done through a previous qualitative analysys step.

References

[1] W.G. Schneeweiss. *The Fault Tree Method*. LiLoLe Verlag, 1999.

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

[3] A. Bobbio, G. Franceschinis, R. Gaeta, and L. Portinale. Parametric fault-tree for the dependability analysis of redundant systems and its high level Petri net semantics. *IEEE Transactions Software Engineering*, 29:270–287, 2003.

[4] A. Bobbio and D. Codetta Raiteri. Parametric fault-trees with dynamic gates and repair boxes. In *Proceedings of the Annual Reliability and Maintainability Symposium*, pages 459–465, Los Angeles, CA USA, January 2004.

[5] Y. Dutuit and A. Rauzy. A linear-time algorithm to find modules of fault tree. *IEEE Transactions on Reliability*, 45:422–425, 1996.

[6] A. Anand and A. Somani. Hierarchical analysis of fault trees with dependencies, using decomposition. *Proceedings IEEE Annual Reliability and Maintainability Symposium*, pages 69–75, 1998.

[7] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic Well-Formed coloured nets and multiprocessor modelling applications. In K. Jensen and G. Rozenberg, editors, *High-Level Petri Nets. Theory and Application*. Springer Verlag, 1991.

[8] G.Franceschinis M.Gribaudo M.Iacono N.Mazzocca V.Vittorini. Towards An Object Based Multiformalism Multi-Solution Modeling Approach. *Proceedings Second Workshop on Modeling of Objects, Components and Agents*, 2002.

[9] V.Vittorini G.Franceschinis M.Gribaudo M.Iacono C.Bertoncello. DrawNet++: a Flexible Framework for Building Dependability Models. *Proceedings International Conference on Dependable Systems and Networks*, 2002.

[10] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Performance Evaluation*, 24:47–68, November 1995.

[11] R.A.Sahner K.S.Trivedi A.Puliafito. *Performance And Reliability Analysis Of Computer Systems; An Example-Based Approach Using the SHARPE Software Package*. Kluwer Academic Publishers, 1996.

[12] D. Codetta Raiteri, G. Franceschinis, M. Iacono, and V. Vittorini. Repairable fault tree for the automatic evaluation of repair policies. *DSN Performance and Dependability Symposium (to appear)*, July 2004.

[13] Z. Tang and J.B. Dugan. Minimal cut set/sequence generation for dynamic fault trees. In *Procedings of the Annual Reliability and Maintainability Symposium*, Los Angeles, CA USA, January 2004.