

Automatically Translating Dynamic Fault Trees into Dynamic Bayesian Networks by Means of a Software Tool

S. Montani, L. Portinale, A. Bobbio, D. Codetta-Raiteri
Dipartimento di Informatica, Università del Piemonte Orientale
stefania.portinal,bobbio,raiteri@mfn.unipmn.it

Abstract

This paper presents a software tool allowing the automatic analysis of a Dynamic Fault Tree (DFT) exploiting its conversion to a Dynamic Bayesian Network (DBN). First, the architecture of the tool is described, together with the rules implemented in the tool, to convert dynamic gates in DBNs. Then, the tool is tested on a case of system: its DFT model and the corresponding DBN are provided and analyzed by means of the tool. The obtained unreliability results are compared with those returned by other tools, in order to verify their correctness

1. Introduction

The modeling possibilities offered by Fault Trees (FT), one of the most popular techniques for dependability analysis of large, safety critical systems, can be extended by relying on Bayesian Networks (BN) [1], [2], [3], [4], [5]. This formalism allows to relax some constraints which are typical of FT. In addition, BN allow to represent local dependencies and to perform both predictive and diagnostic reasoning.

In [6], we have shown how BN can provide a unified framework in which also Dynamic FT (DFT) [7], a rather recent extensions able to treat complex types of dependencies, can be represented.

In particular, dynamic Fault Trees (DFT) introduce four basic (dynamic) gates: the warm spare (WSP), the sequence enforcing (SEQ), the probabilistic dependency (PDEP) and the priority AND (PAND). WSP are dynamic gates modeling one or more principal components that can be substituted by one or more backups (spares), with the same functionality (Fig. 1(a)). The WSP gate fails when the number of operational powered spares and/or principal components is less than the minimum required. Spares can fail even while they are dormant, but the failure rate of an unpowered spare is lower than the failure

rate of the corresponding powered one. More precisely, being λ the failure rate of a powered spare, the failure rate of the unpowered spare is $\alpha\lambda$, with $0 \leq \alpha \leq 1$ called the dormancy factor. Spares are more properly called “hot” if $\alpha=1$ and “cold” if $\alpha=0$.

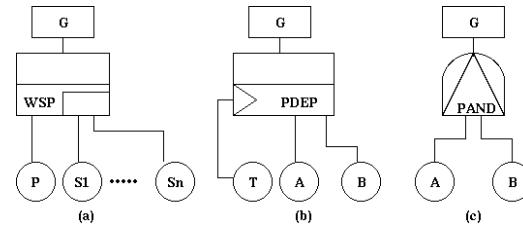


Figure 1: dynamic gates in a DFT.

A SEQ gate forces its inputs to fail in a particular order: when a SEQ is found in a DFT, it never happens that the failure sequence takes place in different orders. SEQ gates can be modeled as a special case of a cold spare [8], so they will not be considered any more in the following¹.

In the PDEP gate (Fig. 1(b)), one trigger event T causes other dependent components to become unusable or inaccessible with probability $pdep \leq 1$. In particular, when the trigger event occurs, the dependent components will fail with the specified probability. The separate failure of a dependent component, on the other hand, has no effect on the trigger event. PDEP has also a non-dependent output, that simply reflects the status of the trigger event.

Finally, the PAND gate reaches a failure state iff all of its input components have failed in a preassigned order (from left to right in graphical notation). While the SEQ gate allows the events to occur only in a preassigned order and states that a different failure sequence can never take place, the PAND does not

¹ The conceptual difference between the two kind of gates is that the inputs to a SEQ do not need to be a component and its set of spares, but can be components covering any kind of function in the FT.

force such a strong assumption: it simply detects the failure order and fails just in one case (in Fig. 1(c) a failure occurs iff A fails before B, but B may fail before A without producing a failure in G).

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

In order to alleviate this limitation, as stated above, we have proposed a translation of the DFT into a Dynamic Bayesian Network (DBN).

With respect to CTMC, the use of a DBN allows one to take advantage of the factorization in the temporal probability model. As a matter of fact, the conditional independence assumptions enables a compact representation of the probabilistic model, allowing the system designer or analyst to avoid the complexity of specifying and using a global-state model (like a standard Markov Chain) when the dynamic module of the considered FT is significantly large.

In this paper, we describe a tool we have implemented to realize an automatic translation of a DFT into the corresponding DBN. We also show an application of the tool functionalities to a real world example.

2. The tool

2.1. Tool functionalities

The tool we have implemented is called DBNet and allows the user to: (1) edit a (dynamic) Bayesian network and draw inferences on it; (2) edit a dynamic fault tree, and (3) automatically convert a DFT into the corresponding DBN, on which both predictive and diagnostic inference can then be drawn. Items (1) and (2) are described below; item (3), which represents the core contribution of this paper, is extensively treated in sections 2.2 and 2.3. The tool architecture is depicted in Fig. 2.

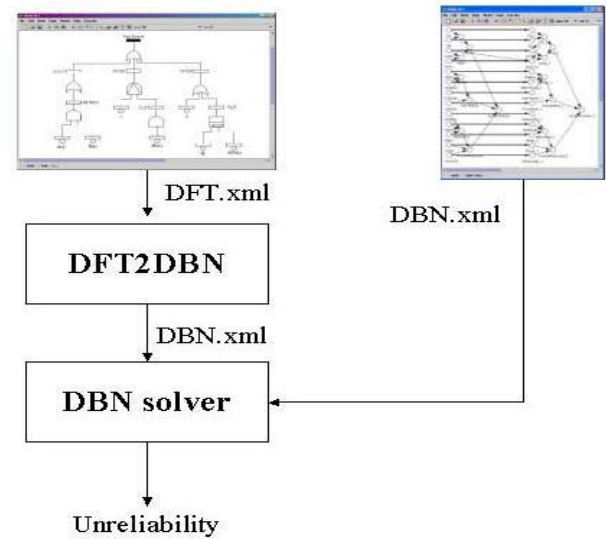


Figure 2: the tool architecture.

2.1.1. Editing a dynamic Bayesian network

By means of the tool graphical interface, which has been built relying on the DrawNet graphical tool [14], the user is allowed to draw the intra-slice conditional dependency model at time t , to duplicate it at time $t+\Delta t$, and finally to provide the inter-slice conditional dependency model. The term Δ is the discretization step, and can be set by the user through the graphical interface. The Conditional Probability Tables (CPT) can be easily inserted relying on a user friendly functionality, in which every row is automatically completed by calculating the last entry as the difference between 1 and the sum of the other values. Additionally, it is possible to identify query nodes (not necessarily the Top Event (TE) – i.e. the global system failure, as usually required in FT analysis), and to provide a stream of observations (i.e. evidences), each one coupled with its observation time. When the network has been fully characterized, the user can activate a *monitoring* procedure, i.e. ask for the calculation of the probability of the query node(s) from the initial time t_0 to the mission time t_m , given a certain discretization step, and given the observation stream – if available. The calculation of the TE is a special case of monitoring, with an empty stream of observations. In addition to the output of classical FT analysis, on the Bayesian network *smoothing* can be required as well. In this case, the past history of the system is rebuilt, on the basis of the stream of observations. This procedure allows to obtain diagnostic knowledge (e.g. given that the TE has been

observed, the probability of failure of basic components is queried). Algorithms for monitoring and smoothing on DBN [12] have been implemented by resorting to Intel PNL (Probabilistic Networks Library), a set of open-source C++ libraries (see <http://www.intel.com/research/mrl/pnl>), to which we have provided some minor adjustments. PNL includes two classes of algorithms: the first class is composed by exact algorithms, based on the calculation of the junction tree; the second class includes parameterized algorithms, which require to cluster the overall network in a set of subprocesses. Subprocesses are then solved independently, thus speeding up the computation time. The algorithms provide the exact result if the clusters have been correctly identified, i.e. if every subprocess evolves independently of the others; an algorithm to automatically find the correct clusters could be implemented as well.

2.1.2. Editing a dynamic Fault Tree

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

DrawNet already included a DFT editor. We have ameliorated it by providing the possibility of drawing dynamic gates, and of indicating query variables and observed variables. On the DFT, the user can also specify the discretization step, as well as the mission time, and the inference algorithm to be adopted on the corresponding DBN. This information is directly inherited by the corresponding DBN when translation is required. In this way, the DFT is exploited as an easy and well known formalism, to which the user is typically already familiar, through which all the needed data for DBN inference can be given in input.

2.2. Translating dynamic gates

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

In a DFT, different configurations of warm spares can be designed. Of particular interest are those in which the same pool of spares is shared across a set of WSP. In this case, each principal component is allowed to request the items in the pool in a precise

order - if more than one is still dormant. As an example, let us consider a situation where two components A and B can be substituted by two spares S1 and S2. In particular, S1 is B's spare, and will substitute A only if: (i) B is working, and (ii) S2 is failed. If B fails, it will request the activation of S1, and only if it is unavailable it will activate S2. S2 is A's spare: analogous considerations hold. Every gate fails iff its principal component and all the available (i.e. working and dormant) spares in the pool fail. The DBN corresponding to this situation is shown in Fig. 3. Note that the network complexity and the CPT dimensions would significantly increase as the number of shared spares increases.

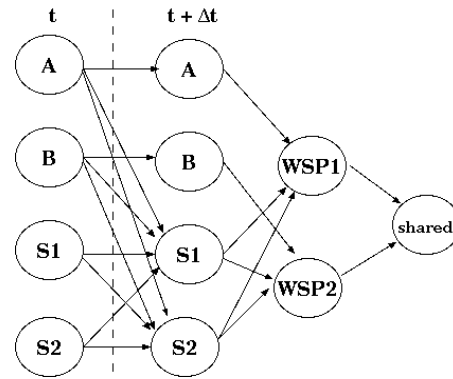


Figure 3: DBN for the WSP gate.

It can be observed that each component node at time $t+\Delta t$ depends on its copy at time t (we consider persistence of faults). Moreover, each spare depends on the two principal components, and on the other spare. Each spare is modeled as a stochastic variable assuming four values, namely: dormant, operative on A, operative on B and failed. If both principal components are working, each spare maintains a failure rate equal to $\alpha\lambda$ at time $t+\Delta t$. On the other hand, if B is down, S1 switches to a failure rate equal to λ (since the spare is now in the active mode); the same happens if B is working, but A and S2 are both failed. S2 works dually on its principal component A. Each WSP gate (in the example, the one having A as its principal component, and the one having B as its principal component) is modeled as a deterministic AND node among its three inputs: the principal components and the two spares in the pool.

2.3. Combining the modules into a single DBN

The three gates we have examined in the previous section can be connected, in order to build a complex

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

In particular, according to [8]:

1. the dependent events of a PDEP can only be basic events, which could be the input of another dynamic gate;
2. WSP can have only basic events as an input, and two or more spare gates can share some spare components; a set of WSP sharing a pool of spares are treated as a single module for translation (see section 2.2);
3. PAND can have basic events or spare gates as an input; two or more PAND can also be combined in a cascade manner.

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

Consider a pair of dynamic gates G1 and G2. Let B be the output of G1 (or an event triggered by G1 if G1 is a PDEP), and an input to G2 (B is therefore the element connecting the two gates). The algorithm works as follows:

1. generate independently DBN1 and DBN2 for G1 and G2 respectively;
2. connect the two networks in correspondence of the nodes they share; when new arcs enter a node, an adjustment to its CPT is required, as follows:
 - a. add all the parents derived from DBN1 and DBN2 as columns in the new CPT;
 - b. in every entry of the table, set the probability of failure of the node to the maximum between the corresponding entries of the CPT in DBN1 and in DBN2. In this way the strongest effect on the probability of failure of the shared component is always considered.
3. repeat the above step for every shared node between the two networks (more nodes could be shared e.g. when G1 is a PDEP, and it triggers more than one input to G2).

3. An example

The example we report is inspired from [15] and consists of a multiprocessor computing system; it is composed by two computing module: CM1 and CM2. Each of them contains one processor, one memory and two hard disks: a primary and a backup disk. In the case of CM1, they are respectively indicated as P1, M1, D11, D12. In the case of CM2, they are: P2, M2, D21, D22 (see Fig. 4).

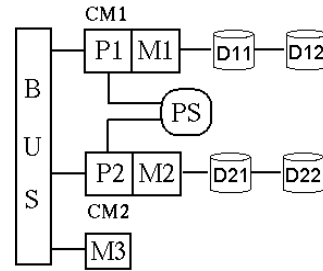


Figure 4: the scheme of the multiprocessor computing system.

Initially, the primary disk is accessed by the computing modules to store and retrieve programs and data, while the backup disk contains the copy of the information inside the primary disk, and is accessed only periodically for update operations. For this reason, in this situation, the failure rate of the backup disk is lower than the failure rate of the primary one. If the primary disk fails, it is replaced in its function by the backup disk; from this moment, the failure rate of the backup disk is equal to the failure rate of the primary one. In other words, the backup disk of a computing module, is the spare of the primary disk.

A spare memory (M3) is present in the system; its aim is replacing M1 or M2 in the case of failure. Moreover, the computing modules are connected by means of the bus N. Finally, both P1 and P2 are functionally dependent on the power supply PS; this means that the failure of PS forces the failure of P1 and P2.

The corresponding DFT is described in figure 5, where a PDEP and four WSP can be recognized.

The time to fail of any component in the system is a random variable ruled by the negative exponential distribution; Tab. 1 shows the failure rate of every component, together with the dormancy factors of the spare components.

Fig. 5 has been drawn by means of the graphical interface of our tool, described in section 3.1.

The DBN corresponding to the DFT in Fig. 5, is shown in Fig. 6 and can be automatically generated given the DFT model, by using our tool.

The nodes inside the DBN derive from the translation of the basic events and of the gates (both static and dynamic) inside the DFT; the translation of the internal events and of the top event, is not necessary [1]. The DBN nodes are duplicated for each time slice ($t, t+\Delta t$) according to the 2TBN representation [12]. Temporal arcs (drawn as thicker lines) connect the nodes representing the temporal copies of the same basic event; in this way, we model the failure persistence allowing the possibility of representing also repair processes.

After the conversion of the DFT in a DBN, we can perform the analysis of the latter by means of our tool. Tab. 2 shows the unreliability of the system versus the mission time varying between 1000 and 5000 hours ($\Delta t = 1$ h), obtained by monitoring the node TE_1 without the observation stream for each node; this method is equivalent to the prediction. The obtained results have been successfully verified by comparison with the results returned by other tools run on the same DFT model. Such tools are DRPFTproc [16] (based on modularization [9] and conversion to Stochastic Petri Nets of dynamic gates) and Galileo [17] (based on modularization, Binary Decision Diagrams (BDD) and CTMCs). Tab. 2 reports also the results obtained using such tools.

Table 1: the failure rates and the dormancy factors.

Component	failure rate (λ)	dormancy factor (α)
N	$2.0E-9$ 1/h	
$P1, P2$	$5.0E-7$ 1/h	
PS	$6.0E-6$ 1/h	
$D11, D21$	$8.0E-5$ 1/h	
$D12, D22$	$8.0E-5$ 1/h	0.5
$M1, M2$	$3.0E-8$ 1/h	
$M3$	$3.0E-8$ 1/h	0.5

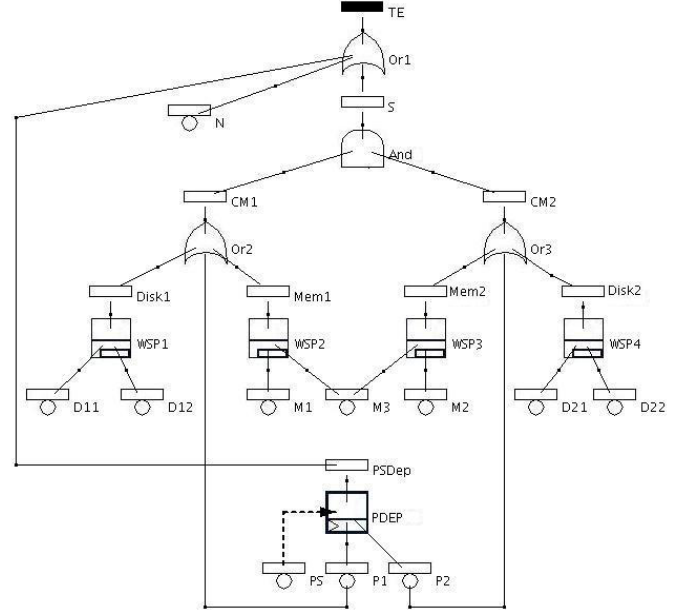


Figure 5: the DFT model of the multiprocessor system.

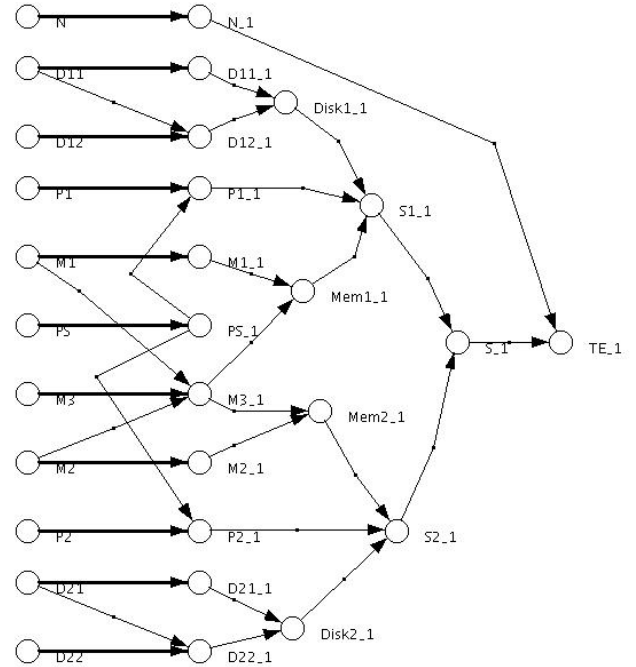


Figure 6: the DBN corresponding to the DFT in Fig. 5.

Table 2: the unreliability results.

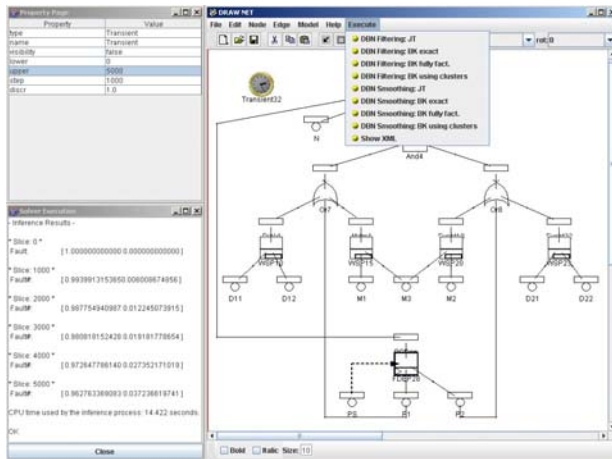
time	DBNet	DRPFTproc	Galileo
1000 h	0.0060086	0.0060088	0.0060088
2000 h	0.0122452	0.0122455	0.0122455
3000 h	0.0191820	0.0191832	0.0191832
4000 h	0.0273523	0.0273548	0.0273548
5000 h	0.0372379	0.0372413	0.0372413

3.1. Graphical interface description

Fig. 7 shows a screenshot of the graphical interface (DrawNet) of our tool. It is mainly composed by three windows; the main window allows the user to draw the DFT model, while in the window named Property Page, it is possible to set the attributes of the node currently selected in the main window. From the Execute menu of the main window, the user can run the conversion and the analysis of the DFT model. At the end of such process, the obtained results are displayed in the window called Solver Execution.

4. Conclusions

In this paper, we have described a tool that allows the user to draw a DBN and to ask for diagnostic or predictive inference on it, as well as to draw a DFT, obtain an automatic conversion into the corresponding DBN, and ask for inference calculation.

**Figure 7: a screenshot of the tool.**

To test both the proposed conversion methodology and the tool performances, we have run some examples, one of which has been presented in this paper: the results obtained using the DBN are basically identical to the ones obtained using other analysis techniques described in the literature.

Our experimental results therefore demonstrate how DBN can be safely resorted to if a quantitative analysis of the system is required.

In the future, we plan to extend the tool capabilities, by adding ad hoc structures to the DFT, which can then be naturally characterized in the corresponding DBN: for example, we will allow the insertion of multi-valued nodes, and the specification of conditional dependencies among basic events. Moreover, we plan to automatically calculate the clusters to be exploited by the approximated inference algorithms, by a modularization procedure on the DFT [9].

In this way, the tool will offer the possibility of drawing the system model relying on the well known and more diffused FT language, and of analyzing the model itself by means of the more powerful DBN.

5. References

- [1] A. Bobbio, L. Portinale, M. Minichino, E. Ciancamerla, "Improving the analysis of dependable systems by mapping fault trees into bayesian networks", *Reliability Engineering and System Safety*, 71:249-260, 2001.
- [2] P. Weber, L. Jouffe, "Reliability modelling with dynamic bayesian networks", *Proc. SAFEPROCESS 2003*, Washington D.C., 2003.
- [3] H. Boudali, J. Bechta Dugan, "A new Bayesian network approach to solve dynamic fault trees", *Proc. RAMS 05*, Alexandria, 2005.
- [4] H. Langseth, "Bayesian networks with application to reliability analysis", *PhD thesis*, Dept. of Mathematical Sciences, Norwegian University of Science and Technology, (<http://www.math.ntnu.no/~helgel/thesis>), 2002.
- [5] J. G. Torres-Toledano, L. E. Sucar, "Bayesian networks for reliability analysis of complex systems", pages 195-206, *Springer Verlag LNCS*, Vol. 1484, 1998.
- [6] S. Montani, L. Portinale, A. Bobbio, "Dynamic bayesian networks for modeling advanced fault tree features in dependability analysis", *Proc. ESREL 2005*, Tri City, June 2005.
- [7] J. Bechta Dugan, S. J. Bavuso, M. A. Boyd, "Dynamic fault-tree models for fault-tolerant computer systems", *IEEE Transactions on Reliability* 41:363--377, 1992.
- [8] R. Manian, D. W. Coppit, K. J. Sullivan, J. Bechta Dugan, "Bridging the gap between systems and dynamic fault tree models", In *Proceedings IEEE Annual Reliability and Maintainability Symposium*, pages 105--111. IEEE Computer Society Press, Washington, DC, 1999.
- [9] R. Gulati, J. B. Dugan, "A Modular Approach for Analyzing Static and Dynamic Fault Trees", *Proc. of the Annual Reliability and Maintainability Symposium*, pages 1-7, 1997.
- [10] T. Dean, K. Kanazawa, "A model for reasoning about persistence and causation", *Computational Intelligence* 5(3): 142-150, 1989.

- [11] P. Dagum, A. Galper, E. Horwitz, "Dynamic network models for forecasting", *Proc. UAI'92*, 1992, 41-48.
- [12] K. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning", *PhD Thesis*, UC Berkley, 2002.
- [13] U. Kjaerulff, "Dhugin: a computational system for dynamic time-sliced bayesian networks", *International Journal of Forecasting* 11:89-101, 1995.
- [14] G. Franceschinis, M. Gribaudo, M. Iacono, N. Mazzocca, V. Vittorini, "DrawNet++: Model Objects to Support Performance Analysis and Simulation of Complex Systems", *Proc. TOOLS 2002*, pages 233-238, Springer Verlag LNCS Vol 2324, 2002.
- [15] M. Malhotra, K. Trivedi, "Dependability modeling using Petri Nets", *IEEE Transactions on Reliability*, R-44:428-440, 1995.
- [16] A. Bobbio, D. Codetta-Raiteri, "Parametric Fault Trees with Dynamic Gates and Repair Boxes", *Proc. of the Annual Reliability and Maintainability Symposium*, pages 459-465, Los Angeles, January 2004.
- [17] K. J. Sullivan, J. Bechta Dugan, D. Coppit, "The Galileo Fault Tree Analysis Tool", *Proceedings of IEEE International Symposium on Fault Tolerant Computing*, FTCS-29, 1999.
- [18] Y. Dutuit, O. Lemaire, A. Rauzy, "New Insight on Measures of Importance of Components and Systems in Fault Tree Analysis", *Proc. of the International Conference on Probabilistic Safety Assessment and Management*, pages 729-734, Universal Academy Press, 2000.