A Generalization of Dynamic Fault Trees through Boolean logic Driven Markov Processes (BDMP)®

M. Bouissou
EDF R&D, MRI department, Clamart, France

ABSTRACT: Dynamic Fault Trees (DFT) and BDMP are both models resembling fault trees, whose function is to specify continuous time Markov chains. Their purpose is to allow the modelling of systems for which the assumption of independence of components does not hold, thus making impossible the use of standard fault tree analysis. While DFT are limited to non repairable systems and take into account only a few specific kinds of dependences, BDMP can model repairable systems and a large variety of dependences. Moreover, BDMP have mathematical properties which allow a drastic reduction of combinatorial problems inherent to the use of Markov chains.

1 INTRODUCTION

Conventional, or "static" fault trees are not at all suited for modeling systems in which there are strong dependencies between components. The assumption of components independence is precisely what makes fault trees so powerful (in terms of size of the systems that can be assessed with this method), but this assumption is extremely restrictive, and may prove to be totally unrealistic and lead to grossly erroneous results for some kinds of systems.

In order to be able to model component dependencies, one has to recur to dynamic models. The most popular are Markov processes, because of their numerous nice mathematical properties. In practice, the direct use of Markov processes has virtually been given up, to be replaced by some higher level formalisms that enable the automatic generation of a (potentially huge) continuous time Markov chain (CTMC).

Dynamic Fault Trees are one of these formalisms. They were introduced by R. Gulati in (Gulati 1996), and have led to several implementations in tools developed by the University of Virginia (Manian et al 1998), and, more recently, by the University of Piemont (Montani et al 2006). They are also available in the Relex commercial tool.

In DFT, special gates have been introduced, that complement the existing static fault tree gates. The most recent definition of these gates is given in (Montani et al 2006). Thanks to these gates, DFT improve the modeling capability of standard fault trees for **non repairable** systems. But their extension to repairable systems has never been considered so far, and the dependences they allow to model are still limited, in spite of the introduction of four new kinds of gates.

The object of this paper is to prove that BDMP constitute a very powerful generalization of dynamic fault trees, that does not suffer from these two limitations.

BDMP are formally defined and their properties are proved in article (Bouissou, Bon 2003). They are implemented in the KB3 workbench, a set of tools developed by EDF (Bouissou 2005), and have been used in numerous studies of complex systems, especially repairable electrical systems.

The paper is organized as follows: section 2 is a quick recall of the definition of BDMP and their main properties. Section 3 gives the BDMP equivalent for each of the four special gates of dynamic fault trees described above, in order to prove that BDMP are a generalization of DFT. Section 4 contains the BDMP solution of the test-case of article (Montani et al 2006), graphically input, and automatically solved with the tools of the KB3 workbench. Finally, we enrich that test-case by supposing that the system is repairable, and we show that the repairable variant can be solved with no additional effort with these tools.

2 BDMP DEFINITION AND PROPERTIES

Although BDMP may seem similar to dynamic fault trees, they are in fact quite different. Instead of adding new kinds of gates, they assign a new semantics to the traditional graphical representation of fault trees, augmented **only** by a new kind of links (these links are called "triggers" and are represented by dotted arrows) and an equivalent of the so-called PAND gates of DFT.

They enable the analyst to combine conventional fault trees and Markov models in a brand new way**.** BDMP have very interesting mathematical properties, which allow a dramatic reduction of combinatorial problems when they are converted into CTMC for their solving. Moreover, they allow to obtain particularly relevant qualitative information in the form of minimal sequences leading to the occurrence of the top event.

The general idea of BDMP, as suggested by their name, is to associate a Markov process (which represents the behavior of a component or a subsystem) to each leaf of a fault-tree. This fault-tree is the structure function of the system.

What is really new with BDMP is that:

- the basic Markov processes have two "modes", corresponding to the fact that the components/subsystems that they model are required or are in standby (of course, they can also have only one mode, and the meaning of the modes may be different in some cases),
- at any time, the choice of the mode of one of the Markov processes (unless it is independent) depends on the value of a Boolean function of other processes.

An extreme case is when the processes are independent. This corresponds to a fault-tree, the leaves of which are associated to independent Markov processes.

2.1 *The elements of a BDMP*

A BDMP ($\mathcal{F}$, r, T, (P_i)) is made of: a multi-top coherent fault-tree $\mathcal{F}$, a **main top event** r of $\mathcal{F}$, a set T of **triggers**, a set of **"triggered Markov processes"** P_i associated to the basic events (i.e. the leaves) of $\mathcal{F}$, the definition of two categories of states for the processes P_i.

A **trigger** is represented graphically with a dotted arrow. The origin and the target of a trigger can be any gate or basic event of $\mathcal{F}$. However, two triggers must not have the same target. This means that it is sometimes necessary to create an additional gate (like G1 in Figure 1) whose only function is to define the origin of a trigger.

Figure 1 is an example of graphical representation of all the notions of BDMP. In this example, we have a fault-tree with two tops: r (the main one) and G1. The basic events are P1, P2, P3, and P4: they can belong to one of the two standard triggered Markov processes defined below. There is only one trigger, from G1 to G2.

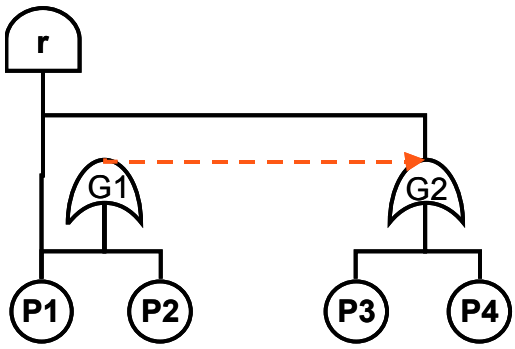


Figure 1. A simple BDMP

2.2 *Definition of a "triggered Markov process"*

Such a process P_i is associated to each basic event i of the fault-tree. P_i is the following set of elements:

$\{Z_0^i(t), Z_1^i(t), f_{0\to1}^i, f_{1\to0}^i\}$

$\{Z_0^i(t), Z_1^i(t)\}$ are two homogeneous Markov processes with discrete state spaces. For $k \in \{0,1\}$, the state space of $Z_k^i(t)$ is A_k^i. For each A_k^i we will need to refer to a part F_k^i of the state space A_k^i. In general, F_k^i will correspond to failure states of the component or subsystem modeled by the process P_i.

$f_{0\to1}^i$ and $f_{1\to0}^i$ are two **probability transfer functions** defined as follows:

- for any $x \in A_0^i$, $f_{0\to1}^i(x)$ is a probability distribution on A_1^i, such that if $x \in F_0^i$, then $\Pr(f_{0\to1}^i(x) \in F_1^i) = 1$
- for any $x \in A_1^i$, $f_{1\to0}^i(x)$ is a probability distribution on A_0^i, such that if $x \in F_1^i$, then $\Pr(f_{1\to0}^i(x) \in F_0^i) = 1$

Such a process is said to be "triggered" because it switches **instantaneously** from one of its modes to the other one, via the relevant transfer function, according to the state of some externally defined Boolean variable, called **"process selector"**. The process selectors are defined by means of triggers. The function of a trigger is to modify the mode of the processes associated to the leaves in the sub-tree under its target when the event that is the origin of the trigger changes from FALSE to TRUE (or conversely). The exact definition of the semantics of a BDMP (in particular when there are several triggers) is too complex to be explained in the present paper, but it can be found in (Bouissou, Bon 2003).

We give in § 2.3 and 2.4 the two standard triggered processes which are most often used in BDMP. Another triggered Markov process, that is very useful for modeling multiphase systems, is depicted in (Bouissou et al. 2005).

2.3 *The warm standby repairable leaf*

This process is used to model a component that can fail both when it is in standby and when it works (this mode corresponds to a process selector equal to 1), but with different failure rates. This component can be repaired whatever its mode. When $\lambda_s = 0$, the model represents in fact a cold standby repairable component, and when $\lambda_s = \lambda$, it represents a hot standby component.

Process 0 Process 1

The transfer functions simply state that when the value of the process selector changes, the component goes from state Standby to Working (or vice-versa) or remains in Failure state with probability 1.

$$f_{0\to1}(S) = \{\Pr(W) = 1, \Pr(F) = 0\},$$
$$f_{0\to1}(F) = \{\Pr(F) = 1, \Pr(W) = 0\}$$
$$f_{1\to0}(W) = \{\Pr(S) = 1, \Pr(F) = 0\},$$
$$f_{1\to0}(F) = \{\Pr(F) = 1, \Pr(S) = 0\}$$

2.4 *The on-demand repairable failure leaf*

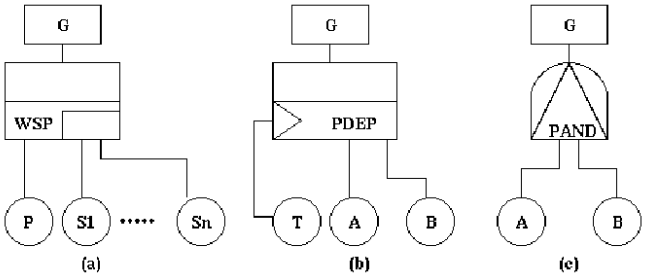
This model is used to represent an on-demand failure, that can happen (with probability γ) when the process selector changes from mode 0 to mode 1.

Process 0 Process 1

$$f_{0\to1}(W) = \{\Pr(W) = 1-\gamma, \Pr(F) = \gamma\},$$
$$f_{0\to1}(F) = \{\Pr(F) = 1, \Pr(W) = 0\}$$
$$f_{1\to0}(W) = \{\Pr(W) = 1, \Pr(F) = 0\},$$
$$f_{1\to0}(F) = \{\Pr(F) = 1, \Pr(W) = 0\}$$

3 BDMP EQUIVALENT OF THE SPECIAL DFT GATES

3.1 *The usual representation of these gates*



(a) (b) (c)

Figure 2. The three dynamic gates with special symbols

In a DFT, SEQ gates, which we examine just below, can be represented by standard symbols for "and", "or", "k/n" gates and a special annotation to indicate that they are sequential. From this point, we will systematically use "k/n" gates whenever we do not want to precise the exact nature of a gate.

3.2 *Equivalent of the **SEQ gate***

The SEQ gate, also known as the Sequence Enforcing gate, forces events to occur in a particular order. The input events are constrained to occur in the left-to-right order in which they appear under the gate. This means that the left-most event must occur before the event on its immediate right, which must occur before the event on its immediate right and so forth.

This is easily represented in a BDMP with the kind of structure described in Figure 3.

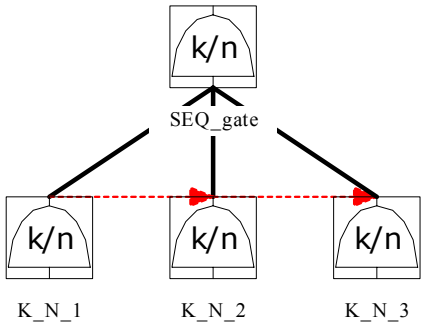
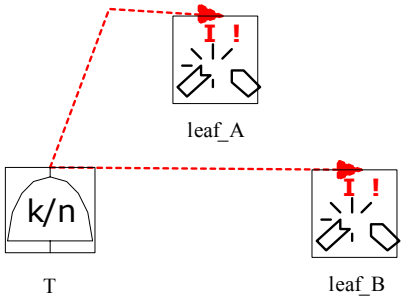


Figure 3. BDMP equivalent to a SEQ gate

Supposing that all basic events of this structure (they are not represented on Figure 3 for a better generality) are of the type described in § 2.3 (with λ_s=0), such a structure forces the event K_N_1 to occur first, because in the initial state, only the events under this gate can be in mode 1. Then when K_N_1 takes the value TRUE, the events under K_N_2 switch to mode 1 (at least, some of them: it depends on the existence of additional triggers in this sub-tree) and thus can take the value TRUE. And so on. This quite simple structure is all what is needed for a non repairable system. For a repairable system, in most cases, a cascade of gates and triggers like the one described in Figure 5, which has a slightly different behavior with regard to reconfigurations after repairs, will be preferred.

3.3 *Equivalent of the **PDEP gate***



T

Figure 4. BDMP equivalent to a PDEP gate

The PDEP gate, also known as the Probabilistic Dependency gate, is not really a "gate", because it is

not involved in the definition of the structure function of the DFT. It has one input called trigger event and can have one or more outputs called dependent events (in Figure 2 (b) the trigger event is called T and the dependent events are A and B).
It is used to indicate that the trigger event causes the dependent events to happen, with a given probability (lower or equal to 1). The BDMP equivalent to such a gate is given in Figure 4.
In this figure, leaf_A and leaf_B must be of the type described in §2.4. The gate T, and the leaves leaf_A and leaf_B of this BDMP excerpt can be attached anywhere in the structure function of the BDMP.

3.4 *Equivalent of **Spare gates and events***

Spare gates and events are used to model cold, warm, and hot spares in the system. A Spare gate has the value TRUE if and only if all its inputs have the value TRUE; in that respect, it behaves like an "and" gate in the structure function. Spare gates are used in conjunction with **Spare events**: a special event type used to model spare usage. In Figure 2, the warm spare gate WSP has a primary input P and several spare event inputs S1…Sn. The meaning of such a structure is that when the failure of the component modeled by P occurs, an available spare is taken from the set of spares to replace it. This can be done several times until all spares are used. Depending on the kind of spare (hot/warm), the failure rate associated to spare events as long as they represent spares in standby can be equal to/lower than the failure rate in function. In the case of cold spares, the standby failure rate is zero. The main interest of Spare gates resides in the fact that spare events can be inputs of several spare gates, thus modeling shared spares. Whenever a spare event has been "used" by a spare gate, it takes the value TRUE for other spare gates.
It is difficult to describe the BDMP equivalent to Spare gates and events without their context when spare events are shared. The reader will find an example in section 4. Assuming that the spare events of the Spare gate of Figure 2 (a) are not shared, the equivalent of that structure (for n =3) would be the sub-BDMP of Figure 5. In this figure, all leaves correspond to the model depicted in 2.3, but the leaf P has a different icon to indicate that its parameter λ_s=0.
If the system is not repairable, this structure can be replaced by a simpler one, similar to the structure depicted in Figure 3. The advantage of this more complicated structure is to ensure that a spare component is in mode 1 if and only if the primary and all spare components with lower numbers are failed. All the spare components can have different characteristics. Depending on the value of their parameter λ_s, they can represent cold, warm or hot spares or any combination of those.

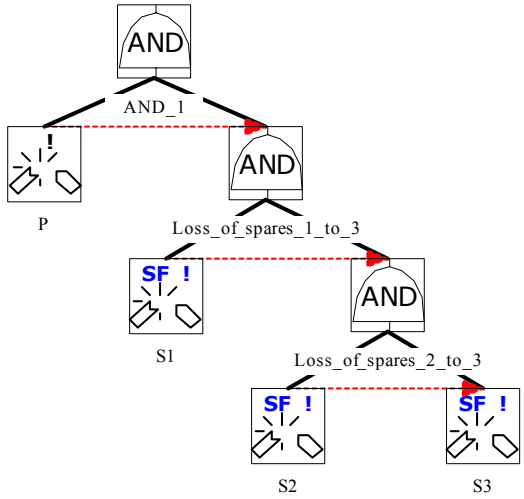


Figure 5. BDMP equivalent to a Spare gate

3.5 *Equivalent of the **PAND gate***

The PAND gate, also known as the Priority AND gate, is used to indicate that the output occurs if and only if all input events occur in a particular order. The difference between a PAND gate and a SEQ gate is that the PAND gate does not **force** its inputs to occur in a given order; these inputs remain independent, and the PAND gate simply **detects** the fact that they have occurred in a given order.

Since BDMP are meant to be models of repairable as well as non repairable systems, we had to take care of what should happen in the case of the repair of components appearing under a gate of that type. We have defined only PAND gates with two inputs *i1* and *i2*, with the following behavior: the PAND gate value changes from FALSE to TRUE whenever *i2* becomes TRUE **while** *i1* is TRUE. As for the change from TRUE to FALSE (step down), a choice must be made by the analyst. Depending on the type of system to be modeled, he may choose one of the following options. The PAND gate steps down when: a) *i1* steps down, b) *i2* steps down, c) the first input (which may be *i1* or *i2*) steps down, d) the last input steps down. Apart from that, the PAND gate of BDMP behaves like a standard "and" gate of a BDMP (in particular with regard to the use of triggers).

If a test on the order of more than two events is needed, the analyst can use a cascade of PAND gates like in Figure 6. In this figure, if no repair is envisaged, the gate PANDi steps up only at the end of a sequence in which K_N_1, K_N_2,… K_N_i+1 step up **in this order**. Many different behaviors can be specified for the effect of repairs, depending on the options chosen for each of the PAND gates.

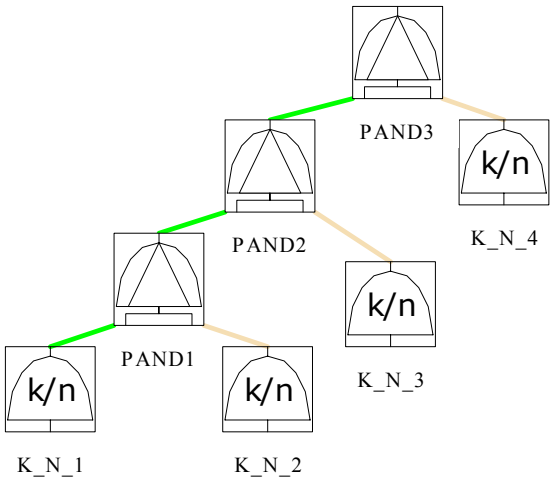


Figure 6. A cascade of PAND gates in a BDMP

4 EXAMPLE: STUDY OF A SYSTEM

4.1 *The system and its models*

In order to show a whole BDMP equivalent to a DFT, we have chosen to solve with the tools of the KB3 workbench the test case given in (Montani et al 2006), because it has already been solved with two methods: the direct transformation of a DFT into a CTMC, and the transformation of the same DFT into a dynamic Bayesian network. Therefore, we will be able to compare our results with these previous resolutions.

Figure 7 gives the physical description of the system to be studied. It is a computing system made of two computing modules, a bus and a shared spare memory M3. The computing module CMi (i=1 or 2) contains one processor Pi, one memory Mi and two hard drives: a primary drive Di1 and a backup Di2.

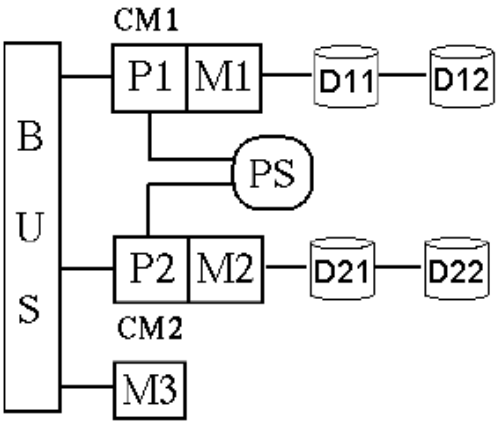


Figure 7. Architecture of the computing system

The backup disks, which are used only from time to time in order to replicate the contents of the primary disks, have a lower failure rate as long as they are in standby (they are warm spares for the primary disks). M3 is a warm **shared** spare for the memories M1 and M2. Its failure rate is also lower when it is in standby. The failure of the bus N or of the power supply PS leads to the loss of the whole system.

Table 1 gives the reliability data for all components. The two first columns of this table are taken from (Montani et al 2006). We have added a column with repair rates to define a variant of the test case, in which all components are repairable.

Table 1. Failure and repair rates of the components

Components	Failure rate in operation λ	Failure rate in standby λ_s	Repair rate μ
N	2.0E-9/h	-	0.1/h
P1, P2	5.0E-7/h	-	0.2/h
PS	6.0E-6/h	-	10/h
D11, D21	8.0E-5/h	-	0.3/h
D12, D22	8.0E-5/h	4.0E-5/h	0.3/h
M1, M2	3.0E-8/h	-	0.1/h
M3	3.0E-8/h	1.5E-8/h	0.5/h

Figure 8 gives a first BDMP modelling the system; this one is easy to build and understand.

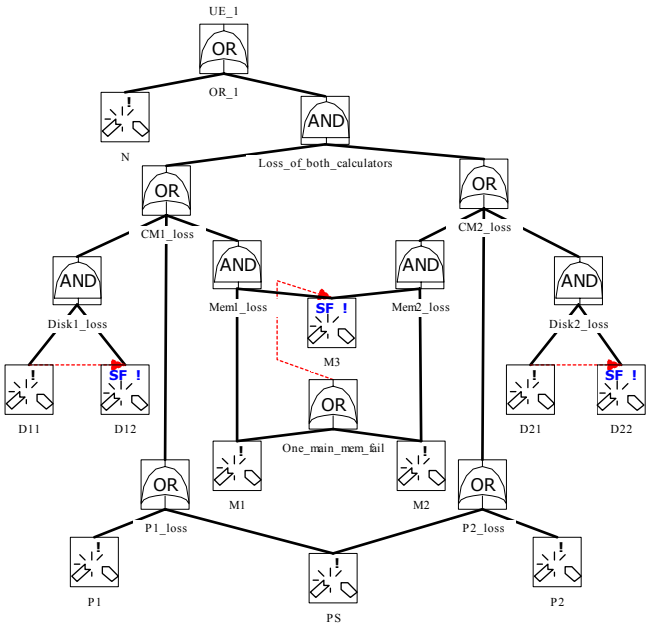


Figure 8. First BDMP modeling the system of Figure 7

However, this first model is not perfectly accurate. Since the leaf M3 (meaning failure of M3) is an input to both gates corresponding to the loss of the memory for the computing modules, this model considers that M3 can replace both M1 and M2, whereas in fact it can replace only *one* of them.

So the idea to model the dependence due to this resource conflict is to represent the memories with a small Petri net (see Figure 10). In order to keep the model simple, we have modeled only the non repairable variant for the memories. In the initial state, the Petri net contains one token in the places named Spares, Mem1_OK, Mem2_OK. As soon as the failure of one of the memories occurs, the corresponding instantaneous transition Replacement is fired, which destroys the token in the Spares place and prevents the other Replacement transition from being fired. Therefore only the first memory failure

can be compensated for by M3. The state of the memory functions is passed to the BDMP via the "Petri leaves" called M1_M3 and M2_M3. For example, M1_M3 is true whenever there is a token in place Mem1_failure, which is tested via the test_1 graphic component, in interaction (not graphically represented) with M1_M3.

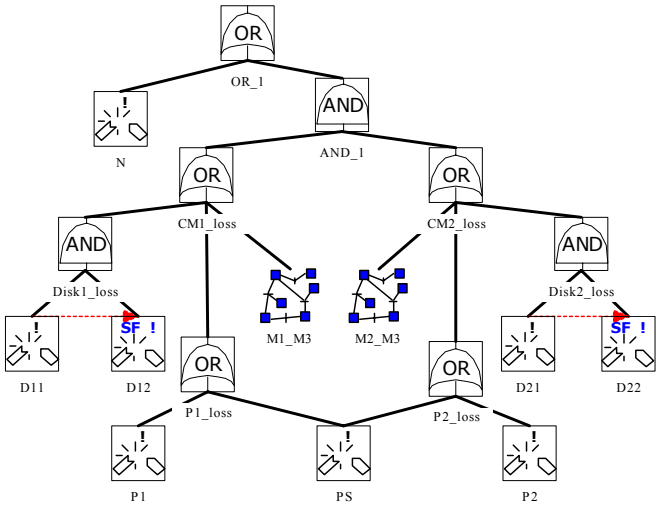


Figure 9. Second BDMP modeling the system of Figure 7

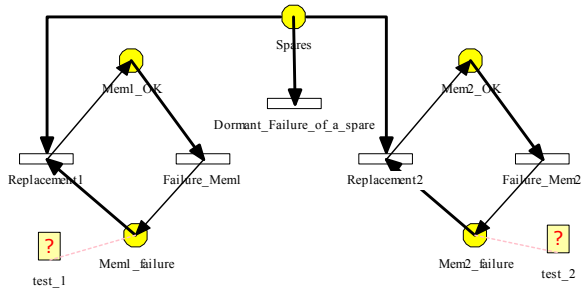


Figure 10. The Petri net part of the second BDMP

4.2 *The results*

Table 2 gives the results we obtained with the above BDMP models, compared with those stated in (Montani et al 2006).

Table 2. Results for the non repairable version of the system

Mission time	Dbnet – in (Montani et al 2006)	Galileo – in (Montani et al 2006)	BDMP + FIGSEQ (model 1)	BDMP + FIGSEQ (model 2)
1000h	6.0086E-3	6.0088E-3	6.00694E-3	6.00694E-3
5000h	3.72379E-2	3.72413E-2	3.72411E-2	3.72411E-2

The relative differences between the various estimations of the system unreliability are under 10^{-3}. However, this is not a real proof of the quality of the different methods, because, as we can see from the output of FIGSEQ in appendix, which gives us the main failure sequences, the failure of the power supply is by far the most important cause of failure of the system. Thus many subtle differences in the modelling or in the probability calculations could be hidden by this large contribution.

Now we can look at the results of the repairable version of the test case.
Because of the imperceptible difference between our two BDMP models, we have chosen the first one to introduce repairs, because all it requires to get a model of the repairable system is to input the values for the repair rates in the leaves of the model (instead of taking them all equal to zero).
By doing so, we could obtain with FIGSEQ approximations (which are also upper bounds) of:

- the asymptotic unavailability of the system: 6.2001E-7
- the unreliability at 5000h: 2.959E-2.

The calculations take a fraction of a second on a standard PC.

Again and in spite of the fact that we took a much shorter repair time for the power supply, it is this component which has by far the largest contribution, both for unreliability and unavailability.

5 CONCLUSION

We have shown in this article that BDMP are more flexible and more general than Dynamic Fault Trees. In particular, they are able to model easily repairable systems. In fact, the only situation in which their use is a little more complex than the use of DFT is when some shared spares must be taken into account. But thanks to the fact that BDMP can embed Petri nets to model special situations, a solution is always possible.

The other important advantages of BDMP, that we could not recall in this article are the fact that they have a formal definition (therefore, whatever its complexity, a BDMP has a well defined semantics) and that they reduce considerably the combinatorial problems one encounters when using Markov chains.

REFERENCES

Bouissou, M. 2005. Automated Dependability Analysis of Complex Systems with the KB3 Workbench: the Experience of EDF R&D. *Proceedings of The International Conference on ENERGY and ENVIRONMENT, CIEM 2005*, Bucharest, (Romania), October 2005.

Bouissou, M., Bon, J.L. 2003. A new formalism that combines advantages of fault-trees and Markov models: Boolean logic Driven Markov Processes *Reliability Engineering and System Safety, Volume 82, Issue 2,* 149-163.

Coppit, D., Sullivan, K.J., Dugan, J.B. 2000. Formal Semantics of Models for Computational Engineering: A Case Study on Dynamic Fault Trees. *Proceedings of the International Symposium on Software Reliability Engineering*, 270-282, San Jose, California, 8-11 October 2000.

Gulati, R.1996. A modular approach to static and dynamic fault tree analysis *Master's thesis*, University of Virginia. Department of Electrical Engineering.

Manian, R., Dugan, J. B., Coppit, D., Sullivan, K. J. 1998. Combining various solution techniques for dynamic fault tree analysis of computer systems *Proceedings of 3rd IEEE International High-Assurance Systems Engineering Symposium*, 21-28.

Montani, S., Portinale, L., Bobbio, A., Varesio, M., 2006. A tool for automatically translating Dynamic Fault trees into Dynamic Bayesian Networks *In: proc RAMS'06*.

Bouissou, M., Dutuit, Y., Maillard, S., 2005. Reliability Analysis of a Dynamic Phased Mission System: Comparison of Two Approaches. *In Modern Statistical and Mathematical Methods in Reliability*, Wilson A., Limnios N., Keller-McNulty S., Armijo Y. (eds) Singapore, World Scientific, 87-104.

Appendix: The preponderant sequences output by FIGSEQ (for the second BDMP, mission time MT=5000h)

Transitions			Proba MT	Aver. Dur. After init	Contrib.
Name	Rate	Class			
[failF OF PS]	6,00E-06	EXP	1,72E-02	0,00E+00	4,63E-01
[failF OF D11]	8,00E-05	EXP			
[failF OF PS]	6,00E-06	EXP	2,92E-03	4,83E+03	7,85E-02
[failF OF D21]	8,00E-05	EXP			
[failF OF PS]	6,00E-06	EXP	2,92E-03	4,83E+03	7,85E-02
[failS OF D12]	4,00E-05	EXP			
[failF OF PS]	6,00E-06	EXP	1,46E-03	4,83E+03	3,93E-02
[failS OF D22]	4,00E-05	EXP			
[failF OF PS]	6,00E-06	EXP	1,46E-03	4,83E+03	3,93E-02
[failF OF D21]	8,00E-05	EXP			
[failF OF D22]	8,00E-05	EXP			
[failF OF D11]	8,00E-05	EXP			
[failF OF D12]	8,00E-05	EXP	5,56E-04	2,42E+04	1,49E-02
[failF OF D11]	8,00E-05	EXP			
[failF OF D12]	8,00E-05	EXP			
[failF OF D21]	8,00E-05	EXP			
[failF OF D22]	8,00E-05	EXP	5,56E-04	2,42E+04	1,49E-02
[failF OF D21]	8,00E-05	EXP			
[failF OF D11]	8,00E-05	EXP			
[failF OF D22]	8,00E-05	EXP			
[failF OF D12]	8,00E-05	EXP	5,34E-04	2,23E+04	1,43E-02
[failF OF D21]	8,00E-05	EXP			
[failF OF D11]	8,00E-05	EXP			
[failF OF D12]	8,00E-05	EXP			
[failF OF D22]	8,00E-05	EXP	5,34E-04	2,23E+04	1,43E-02
[failF OF D11]	8,00E-05	EXP			
[failF OF D21]	8,00E-05	EXP			
[failF OF D22]	8,00E-05	EXP			
[failF OF D12]	8,00E-05	EXP	5,34E-04	2,23E+04	1,43E-02
[failF OF D11]	8,00E-05	EXP			
[failF OF D21]	8,00E-05	EXP			
[failF OF D12]	8,00E-05	EXP			
[failF OF D22]	8,00E-05	EXP	5,34E-04	2,23E+04	1,43E-02