A SYSTEMATIC INCLUSION OF DIAGNOSIS PERFORMANCE IN FAULT TREE ANALYSIS

Jan Åslund [1], **Jonas Biteus** [1], **Erik Frisk** [1], **Mattias Krysander** [1], **Lars Nielsen** [1]

[1] *Department of Electrical Engineering, Linköpings universitet, 581 83 Linköping, Sweden,* `jaasl@isy.liu.se`

Abstract: Safety is of major concern in many applications such as in automotive systems and aerospace. In these applications it is standard to use fault trees, and a natural question in many modern systems that include sub-systems like diagnosis, fault tolerant control and autonomous functions, is how to include the performance of these algorithms in a fault tree analysis for safety. Many possibilities exist but here a systematic way is proposed. It is shown both how safety can be analyzed and how the interplay between algorithm design in terms of missed detection rate and false alarm rate is included in the fault tree analysis. Examples illustrate analysis of diagnosis system requirement specification and algorithm tuning.

Keywords: Safety analysis, redundancy analysis, threshold selection.

1. INTRODUCTION

Safety is of major concern in many applications, and the interest is increasing in safety analysis (Villemeur, 1992). The reason is that new design possibilities have to be evaluated, for example since it is now possible to use diagnosis and thereby analytical redundancy instead of hardware redundancy. This has created a new set of problem formulations to study. One fundamental question is of course if a system becomes safer when a diagnosis function is introduced, and if so, by how much? Another question is how to formulate specification requirements on diagnosis algorithms so that overall system safety is as good as possible. This also naturally leads to the question of how to select internal design parameters in the algorithms. One simple example is that a selection of a threshold balances the rates of missed detection and false alarm, and where to put this balance very much depends on the situation and how it propagates to overall system safety.

To get a handle on these questions it is necessary to have a quantitative method and in this respect fault tree analysis is a natural starting point. It is the basic tool in safety analysis, and may even be a requirement from government, e.g. when declaring air worthiness for aircrafts. Having made this choice, the question is now how to include properties of diagnostic algorithms in fault tree analysis. It should be noted that the main concern here is the interplay between safety and algorithms, and that this should not be confused with the more studied problem on safety of software. It is here assumed that the software is a correct coding of the specified algorithm following the procedures for implementation of safety critical systems.

The purpose of this paper is to put forward the problems described above that to our best knowledge have not been given a treatment previously, and to present a possible solution. In Section 2 fault tree analysis is recapitulated, and in Section 3 diagnosis performance and central concepts like false alarm and missed detection are recalled. It is clear that a fault tree in general, for a certain system, can be formulated in different ways. Nevertheless, in spite of the possible ambiguity in original fault tree formulation, one can look for

[1] Funding from the Swedish National Aeronautics Research Program (NFFP) is gratefully acknowledged.

systematic ways of introducing diagnosis properties in a given fault tree. This is proposed in Section 4 based on algorithm performance in terms of false alarm and missed detection. Further, the use of false alarm rate and missed detection rate is the link to parameter setting of the algorithms, and thus the foundation for both requirements specification on one hand and for algorithm tuning on the other hand, and in Section 5 the proposed methods are applied to these generic examples chosen to illustrate the fundamental questions posed in the beginning of this introduction. Finally, the conclusions are drawn in Section 6.

2. FAULT TREE ANALYSIS

Fault tree analysis is a systematic way to investigate credible causes for an undesired event in a system (Stamatelatos and Vesley, 2002; Vesley *et al.*, 1981). The logical relationships between the undesired event and the basic events that lead to the undesired event are presented in a fault tree.

If the probabilities for the basic events are known, the probability for the top event to occur can be computed. In general there might exist dependencies between the basic events and the same basic event might appear more than once in the fault tree, as will be seen in the examples later on. It is straightforward to treat these dependencies in the calculations and there exists a multitude of fault tree software for this purpose.

Example 1. Fig. 1 shows a fault tree that gives the relationship between the top event *system failure* and the basic events e_1, e_2, and e_3 which are assumed to be independent. The gate symbols denote the relationships between the input events below the gates and the output event above. This fault tree will be used

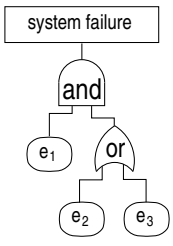


Fig. 1. A fault tree.

in the following sections where it is assumed that the basic events are sensor failures, i.e. event e_i denotes that sensor i is broken, and the probability for this is denoted by p_i. The probability for the top event can easily be computed as

$$P(\text{system failure}) = p_1(p_2 + p_3 - p_2p_3)$$

♦

3. DIAGNOSIS PERFORMANCE

A common way to perform diagnosis is to use a set of tests. Each of these tests consists of a test quantity T_i, and a threshold J_i.

The test quantity T_i, also called *residual*, is designed such that T_i is small if the system to be diagnosed is OK and large otherwise. The test quantity T_i is compared to a threshold J_i and if $T_i > J_i$ then the test is said to alarm. The decision is that the process to be diagnosed is not okay, i.e. that component i is $\neg$OK. In statistical theory (Berger, 1985) the hypothesis "component i is OK" is called the *null hypothesis* of a test and is denoted H_i^0. In (Nyberg, 2002) this statistical theory is included in a diagnosis framework.

To alarm when the supervised system is OK, i.e. H_i^0 is true, is called a *false alarm* (FA_i). Further, to not alarm when the supervised system is faulty, i.e. H_i^0 is false, is called a *missed detection* (MD_i). The probability of these two events define important performance measures of a test as follows. The false alarm probability is

$$P_{\text{FA}_i} = P(T_i > J_i | H_i^0 \text{ true}) \tag{1}$$

and the missed detection probability is

$$P_{\text{MD}_i} = P(T_i \leq J_i | H_i^0 \text{ false}) \tag{2}$$

An ideal test gives no false alarms and no missed detections.

When designing tests, the last step is to compute a threshold. Dropping the index i for now, the FA and MD probabilities are, as can be seen in (1) and (2), functions of the threshold J. A typical example of FA and MD probabilities as functions of the size of the threshold can be seen in Fig. 2. To obtain a small FA probability the threshold must be large, but with a large threshold the MD probability gets large. Hence the choice of threshold adjusts the compromise between a small FA probability and a small MD probability.

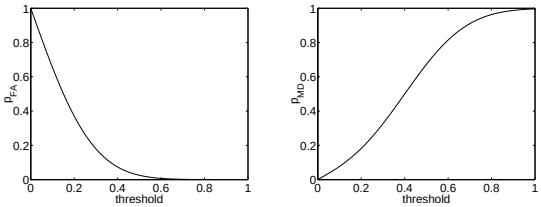


Fig. 2. Probabilities of FA and MD as functions of threshold size for a test.

4. INCLUDING DIAGNOSIS PERFORMANCE IN A FAULT TREE

By including the effects of diagnosis algorithms in fault trees, the consequences of false alarms and missed detections are explicitly modeled. It will later be shown that for example the compromise between

FA and MD can be computed by using the tree including diagnosis. It is described how one test can be included in a fault tree and a generalization to several tests will be straightforward. Before a systematic method to include diagnosis in an existing fault tree will be presented, an illustrative example is given.

Example 2. Consider Example 1 and its fault tree shown in Fig. 1. In order to decrease the system failure probability, a backup system and a diagnosis test are added to the original system.

The system is OK if either the original system or the backup system is switched on and is OK. The backup system consists of a sensor called sensor 4. The backup system is not okay if sensor 4 is *not okay*, denoted e_4. The test supervises if sensor 2 is okay, i.e. $\neg e_2$. If the test alarms then the backup system is turned on and the original system is turned off.

An expanded fault tree where diagnosis and backup system are included is shown in Fig. 3. The tree in Fig. 1 can be found in the left branch of the expanded tree. Original system failure is connected to an and-gate together with no alarm, because original system failure leads only to a system failure in absence of an alarm, i.e. the alarm deactivated tree. The right part of the tree describes the logic when alarming, i.e. the alarm-activated tree. This branch consists of a backup failure tree and the alarm tree A. The important diagnosis events FA and MD are leafs in this tree. ♦

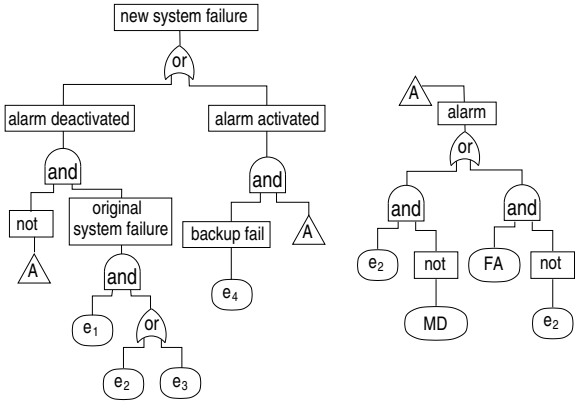


Fig. 3. A fault tree where diagnosis performance is included.

Next a general and systematic inclusion of diagnosis will be described. The fault tree describing the original system will be denoted T_0. The procedure consists of four steps which are illustrated in Fig. 4 and Fig. 5(a)-(c) respectively. The four steps are briefly described as 1) to construct an alarm tree, 2) to construct an alarm-activated tree, 3) to identify the alarm-deactivated event in the original tree, and 4) to insert the alarm-activated tree in the tree obtained in step 3. In general, there can be several alarm-deactivated trees and alarm-activated trees for one test. The following procedure only describes the case with one alarm deactivated tree and one alarm-activated tree. In the case of several such trees, some of the steps has to be performed several times. Next the four-step procedure will be described in detail and the fault tree in Fig. 3 will be used to exemplify each step.

The first step of the inclusion of a test is as said before to make a tree that describes the event *alarm*, i.e. the sub-tree denoted A in Fig. 3. To include the diagnosis performance measures (1) and (2), the alarm event has to be expressed using FA and MD. An alarm can either be a false alarm if H^0 is true or a correct alarm if H^0 is false. By the definition of MD it follows that a correct alarm is equivalent to the event *not missed detection*. Hence, alarm for a specific test can always be expressed as

$$\text{alarm} = (\text{FA} \wedge H^0) \vee (\neg\text{MD} \wedge \neg H^0) \qquad (3)$$

A general way to express this alarm event as a fault tree is shown in Fig. 4. By using this general form of an alarm tree, the remaining task is to model the tree denoted B describing when the null hypothesis H^0 is true.

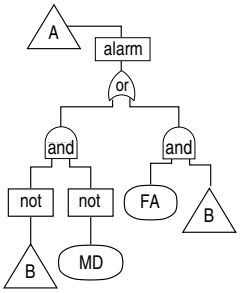
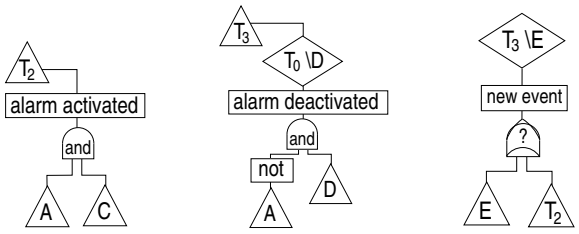


Fig. 4. A general fault tree describing the $alarm$ event. Everything except for the tree denoted B is fixed.

To illustrate this first step for Example 2, the null hypothesis H^0 of the test is $\neg e_2$. If B is substituted in Fig. 4 by $\neg e_2$, the alarm tree in Fig. 3 is obtained.



(a) Step 2 (b) Step 3 (c) Step 4

Fig. 5. Trees constructed in the different steps.

The second step is to make the alarm-activated tree denoted T_2 with structure as in Fig. 5(a). The tree T_2 consists of the alarm tree A obtained in the first step and the sub-tree denoted C. This tree is to be constructed such that it describes events that only affect the failure probability when an alarm has occurred. In Fig. 3, C corresponds to the sub-tree describing backup failure. Since the backup system is turned on

only during an alarm, it can of course only affect the failure probability during an alarm.

The third step is to construct the tree T_3 shown in Fig. 5(b). It consists of a sub-tree D of T_0, A, and $T_0 \backslash D$, where $T_0 \backslash D$ denotes the sub-tree in T_0 induced by the vertices not in D. The tree D is defined as the sub-tree of the original T_0 that affects the failure probability only and just only when no alarm has occurred, i.e. it should not give any affect when the alarm is active. In Example 2, the original fault tree is shown in Fig. 1. Since the entire original system is turned off in the occurrence of an alarm, it follows that the entire original tree is equal to D. This means that $T_0 \backslash D$ in this example is empty and the resulting tree T_3 is *system failure* as top event and below the top event is the left branch of the tree in Fig. 3. Even though e_2 is included in D, it can affect system failure through the alarm tree when alarming. However, since this dependence is handled in A, it should not be considered when identifying D.

As the *fourth step* the trees T_2 and T_3 are merged as shown in Fig. 5(c). In case of an alarm, precautions are taken for one event, e, in the fault tree T_3. The fault tree that describes e in T_3 defines E in Fig. 5(c). The tree E is replaced in T_3 by a *new event* and a gate connecting E and T_2 as shown in Fig. 5(c). The type of the gate is given by the system descriptions. For Example 2, the backup system during an alarm has the same function as the original system during no alarm. This means that e is the event *alarm deactivated* in T_3. From the system description, it follows that the two trees E and T_2 are combined with an or-gate directly below *new event*, that in this case is *new system failure*, resulting in the final tree shown in Fig. 3.

Example 3. Consider again Example 1, but in a new scenario. Let sensor 1 be supervised. If the test alarms, a prediction of the measured value of sensor 1 is used instead. The prediction is based on an observer that uses measurements of sensor 2. The resulting fault tree is shown in Fig. 6 where the four trees A, B, C, and D are encircled. How to obtain this result will next be explained by following the proposed procedure step by step.

Since the test checks if sensor 1 is OK, the null hypothesis H^0 of the test is $\neg e_1$, which defines sub-tree B. The alarm tree A is straightforwardly constructed and step 1 is completed.

When an alarm occurs, a predicted value of the measurement is used. Therefore, the sub-tree C is in this example describing the event *bad prediction*. A bad prediction is a consequence of e_2 or a faulty observer algorithm denoted *observer faulty* in Fig. 6. The resulting alarm-activated tree T_2 is the *active bad prediction*-tree.

To perform step 3, notice that sensor 1 is turned off during an alarm. Hence it is only e_1 in the original tree that does not affect the system failure during an alarm, i.e. $D = e_1$. The alarm deactivated event is denoted *active bad value* in Fig. 6.

In step 4, recall that sensor 1 is predicted in case of an alarm, i.e. a precaution is taken to reduce the risk of *active bad value* of sensor 1, which is the event e for this example. By introducing the event *bad value* combining the obtained trees as the system description specify, the fault tree in Fig. 6 is obtained. ♦

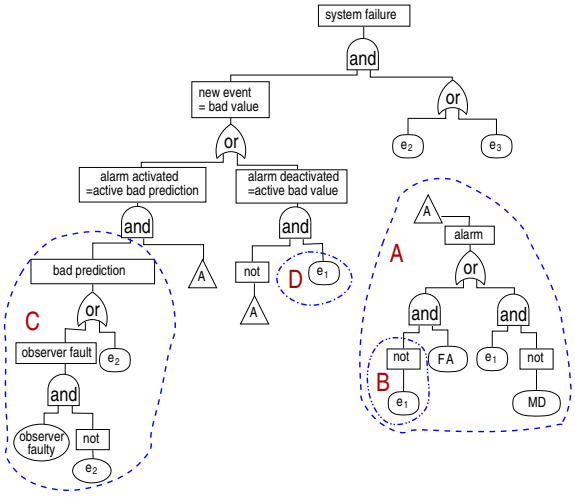


Fig. 6. Tree including diagnosis performance.

4.1 *Simplifications*

In some cases it is possible to make approximations in the calculations of the probability of the top event. The main objective to do so is to obtain simplifications in the fault tree and the following example illustrates how this can be done.

Example 4. Consider the diagnosis and backup system in Example 2 with the top event system failure (SF) as top event. One way to calculate the probability for the top event in this case is

$$\begin{aligned} P(\text{SF}) =& P(\text{SF}|\neg\text{alarm})P(\neg\text{alarm}) \\ &+ P(\text{SF}|\text{alarm})P(\text{alarm}) \end{aligned}$$

where the probabilities $P(\neg\text{alarm})$ and $P(\text{alarm})$ can be computed as before. The conditional probabilities $P(\text{SF}|\neg\text{alarm})$ and $P(\text{SF}|\text{alarm})$ can be calculated using the fault trees for original system and backup system respectively, with the original probabilities replaced by conditional probabilities $P(e_i|\neg\text{alarm})$, $i = 1, 2, 3$ and $P(e_4|\text{alarm})$. The probabilities for the events e_1, e_3, and e_4 are not affected by the conditioning since they are independent of the supervised event e_2, and

$$P(e_2|\neg\text{alarm}) = \frac{P(\neg\text{alarm}|e_2)P(e_2)}{P(\neg\text{alarm})}$$

In many applications the alarm frequency is low and thus $P(\neg\text{alarm}) \approx 1$. This approximation and the definition $P(\neg\text{alarm}|e_2) = P_{\text{MD}}$ give

$$P(e_2|\neg\text{alarm}) \approx P_{\text{MD}}P(e_2)$$

Fig. 7 shows how these approximations can be represented in the fault tree, where the alarm tree is the same as before. ♦

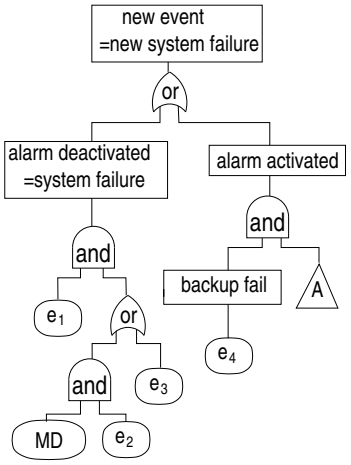


Fig. 7. Approximate fault tree.

The benefit from this way to include the diagnosis performance is that the influence of the diagnosis system is presented close to the supervised event. This can be desirable when the fault tree is large and difficult to survey. However, in more complicated cases, like the one considered in Example 3, the conditional probabilities can be more difficult to compute and to interpret.

5. GENERIC ILLUSTRATIVE EXAMPLES

Previous sections have described the relation between performance of diagnosis algorithms and overall system safety. Here, a few examples are presented on how this relation can be used.

5.1 *Requirements Specification*

Suppose external requirements state numerical demands on the probability for the top event, e.g. the probability for a system failure, and it has been concluded that this requirement can not be fulfilled without a diagnosis system. Then a question is what performance requirements on the diagnosis system are necessary to ensure that the overall requirement is fulfilled.

As described in Section 3, the specifications are condensed into the probability for false alarm, P_{FA}, and missed detection, P_{MD}, for each diagnosis test. In this analysis, no specific algorithm is considered, therefore, the performance of an algorithm is here specified in a diagram with P_{MD} and P_{FA} on the axes. Fig. 8 shows the performance of the diagnosis algorithm from Section 3. The curve is parameterized by the threshold and can be directly obtained from Fig. 2. Now, the overall requirement on the top event is de-

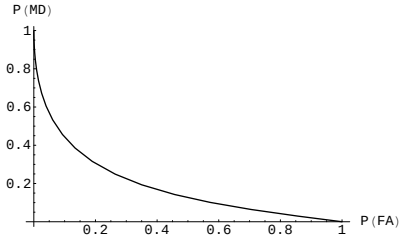


Fig. 8. The curve indicates performance of a diagnosis algorithm in a $P_{\text{FA}}/P_{\text{MD}}$ diagram.

fined as an upper limit of the probability (or a function of the probability) for the top event. Therefore, the overall performance requirement is stated as

$$P(\text{top event}) = f(P_{\text{MD}}, P_{\text{FA}}) \leq \beta \qquad (4)$$

where the function f is given by the fault tree, see Section 2, and β is the performance specification. The requirement specification on the diagnosis algorithm will then be the set of possible pairs $(P_{\text{FA}}, P_{\text{MD}})$ that satisfies (4). The function f in (4) is a low order polynomial in the probabilities P_{FA} and P_{MD} and even linear in the case of only one diagnostic algorithm. It is thus generally possible to obtain a simple parameterization of the solution set to (4).

Example 5. Consider again Example 2, where sensor 2 is supervised. In this example, the probabilities for the basic events are assumed to be $p_1 = 0.1$, $p_2 = 0.005$, $p_3 = 0.01$, $p_4 = 0.005$, where $p_i = P(e_i)$. Assume that the overall requirement is that the probability for the top event must be lower than $\beta = 0.0015$. Performing the analysis outlined above results in the requirement specification on the diagnosis algorithm which is indicated by the shaded area in Fig. 9. In this case, inequality (4) becomes the linear, in P_{MD} and P_{FA}, inequality

$$(1-p_2)(p_4-p_1p_3)P_{\text{FA}} + p_2(p_1-p_4)P_{\text{MD}} + p_1(1-p_2)p_3 + p_2p_4 \leq \beta \qquad (5)$$

The solid line shows the performance of the diagnosis algorithm from Section 3, and since the intersection of the two overlap, there exist a feasible tuning, i.e. threshold selection, of the algorithm that fulfills overall system requirements. ♦

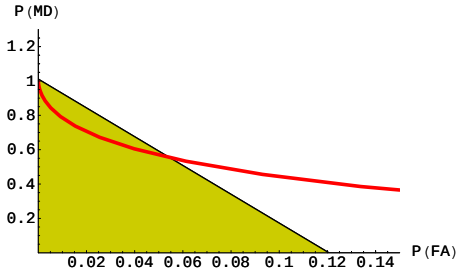


Fig. 9. Feasible diagnosis performance (shaded) and performance of algorithm (curve).

In practice, there are often several top events where each event is represented using a fault tree. Often, these events lead to opposing requirements, e.g. systems safety vs. availability, and there is a need to make a compromise. Extension to the case with more than one top event is straightforward. Apply the procedure above for each tree to obtain a set of feasible performance specifications. Then validate that the intersection of all these sets is non-zero, i.e. that there exist a pair of false alarm/missed detection probabilities that satisfies all top event requirements.

5.2 *Optimal Threshold Selection*

The probability $P(\text{top event})$ can be written as a function of P_{MD} and P_{FA}. Given a diagnosis algorithm, the probabilities P_{MD} and P_{FA} are in their turn functions of the threshold J, as described in Section 3. The probability for the top event can therefore be written as

$$P(\text{top event}) = f(P_{\text{MD}}(J), P_{\text{FA}}(J)) = F(J)$$

In Fig. 10 the function F is shown for the fault tree in Fig. 3 with the diagnosis performance measures from Fig. 2. In this example it is natural to choose J so that $P(\text{top event})$ is as small as possible, i.e. to solve the optimization problem

$$\min_{J \in I} F(J)$$

where I is the set of admissible values for J.

The case with several fault trees with different top events leads to a multi-objective optimization problem with more than one objective function. One way to treat this problem is to multiply these functions by a weight and then add them together to a single objective function.

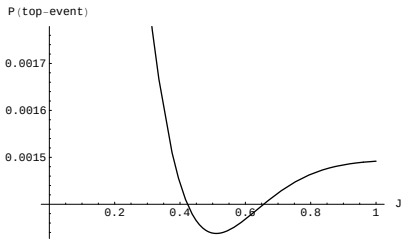


Fig. 10. Probability for the top event as function of the threshold.

6. CONCLUSIONS

There is an increasing number of systems that use advanced control software, and in many of these applications diagnosis systems are integrated to increase availability and safety. To be able to analyze if a system is safe or not, the common approach is to use FTA, and when diagnosis systems are introduced it should be possible to include them in the safety analysis.

A systematic way to include diagnosis in FTA was presented. A fault tree, T_0, for a given system can be formulated in many ways, but nevertheless Section 4 gives a systematic method that in four steps expands a given T_0 to include the diagnosis system. These four steps are straightforward to implement in a computerized tool, and this means that it is possible to compare different diagnosis configurations and parameter settings as was described in Section 5. In Section 5.1 it was shown how requirements on overall system performance can be systematically transferred, via a FTA, into performance requirements on the diagnosis system. Further, in Section 5.2 it was shown how an optimization criterion in a similar straightforward way is obtained, making optimization of parameter tuning simple.

In conclusion, it is believed that a major advantage of the proposed methodology is that it is structured so that it enables tools for interaction regarding the interplay between algorithms and safety, and thus both will result in better systems but also save valuable engineering time.

7. REFERENCES

Berger, James O. (1985). *Statistical Decision Theory and Bayesian Analysis*. Springer.

Nyberg, Mattias (2002). Model-based diagnosis of an automotive engine using several types of fault models. *IEEE Transaction on Control Systems Technology* **10**(5), 679–689.

Stamatelatos, M. and W. Vesley (2002). Fault tree handbook with aerospace applications. Technical report. NASA. U.S.A.

Vesley, W., Goldberg, Roberts and Haasl (1981). Fault tree handbook. Technical Report NUREG-0492. U.S. N.R.C.. U.S.A.

Villemeur, Alain (1992). *Reliability, availability, maintainability and safety assessment*. Vol. 1 & 2. John Wiley & sons. U.K.