

ALGEBRAIC MODELLING OF FAULT TREES WITH PRIORITY AND GATES

Guillaume Merle, Jean-Marc Roussel

LURPA, ENS Cachan
61, avenue du Président Wilson
94235 Cachan Cedex, France
{merle,roussel}@lurpa.ens-cachan.fr

Abstract: This paper presents a formal framework allowing to extend the simplification of static fault trees to fault trees built with gates PRIORITY AND. The laws which make these simplifications possible have been demonstrated thanks to a homogeneous algebraic definition of each gate studied. These definitions use a mathematical model of events able to take into account their order of appearance. The processing of an example points out the possibilities offered by this algebraic framework dedicated to non-repairable faults. *Copyright ©2007 IFAC*

Keywords: Fault trees, minimal cut sets, temporal gates, algebraic approach.

1. INTRODUCTION

A fault tree is a standardized well-known model used in engineering to express graphically what the causes of an undesired/catastrophic event are (IEC 2006) (Vesely *et al.* 2002). A fault tree is often built top-down starting from the undesired event. All the intermediate events which contribute to the undesired event are expanded recursively. The recursion ends with the basic events, which represent the finest level of abstraction chosen for the study. Each analysed event – top or intermediate event – is connected to its causes – sub-events – by a gate. An AND gate indicates that all sub-events are necessary to trigger the analysed event; for an OR gate, only one sub-event is necessary.

The rules of Boolean algebra are commonly applied to restructure fault trees to simpler, equivalent forms. Amongst them, the minimal cut set form allows quantitative and qualitative evaluations to be performed in a straightforward manner (Vesely *et al.* 2002). *Qualitative analysis* identifies single failures or events that alone can cause the

top event to occur. *Quantitative analysis* consists of the determination of top event probabilities and basic event importances.

In the Fault Tree Handbook (Vesely *et al.* 1981), a distinct gate was defined to allow priorities between events: temporal gate PRIORITY AND (PAND). In addition to the condition of gate AND, one input event must occur before the other one for its output event to occur. However, Boolean algebra cannot take into account this sequence of events since events are modelled by Booleans. Consequently, the fault trees using this gate could not be simplified with the laws of Boolean algebra.

The aim of this paper is to define a homogeneous and unambiguous formal framework able to give gate PAND a semantics which would allow the determination of the laws necessary to simplify fault trees. In order to preserve the coherence with the current results, the framework proposed must necessarily give gates OR and AND a semantics allowing to determine the laws commonly used for the calculation of minimal cut sets.

This paper mainly focuses on the formal framework proposed. Section 2 deals with the detailed presentation of the problem and the hypotheses made. The formal framework, described in section 3, relies on the mathematical definition of the elements present in a fault tree: events, gates, ... This definition allowed us to establish the laws necessary to the simplification of fault trees. The application of these laws is illustrated in section 4.

2. PROBLEM

In this paper, the term *fault* is used to refer to fault tree events in order to avoid the confusion between the concept of event used in fault trees and the concept used in discrete event systems.

Let us consider the fault tree in figure 1. Its top fault is s , its basic faults are a, b, c, d , and its intermediate faults are m, n, p, q . This fault tree contains five gates – three OR gates, one AND gate and one PAND gate – whose definitions proposed in the Fault Tree Handbook (Vesely *et al.* 1981) are recalled in table 1.

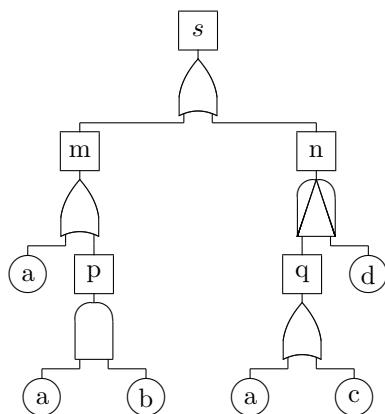


Fig. 1. *Fault Tree with a temporal gate*

Under this form, this fault tree cannot be directly exploited due to redundant parts. When fault trees are *static*, i.e. trees only containing OR and AND gates, the only knowledge of the state of the basic faults – failing or not – is sufficient to simplify these trees. For the tree in figure 1, sub-tree m can be reduced to leaf a by using the following simplification law: $a + (a.b) = a$. When fault trees are *dynamic*, e.g. trees containing temporal gates, only knowing the state of the faults is no longer sufficient since the gates also refer to the order of appearance of the faults – cf. definition 3 of table 1. As the Boolean model of a fault does not contain this kind of information, the classical laws of Boolean algebra cannot produce a solution.

Much work has been carried out to give an operational semantics to temporal gates. Some authors convert temporal fault trees into state-based models to determine the basic fault sequences – or minimal cut sequences, as suggested in (Tang and

Symbol	Definition from (Vesely <i>et al.</i> 1981)
OR 	The output fault occurs only if one or more of the input faults occur. from figure IV-2
AND 	The output fault occurs only if all the input faults occur. from figure IV-5
PRIORITY AND 	 from figure IV-12

Table 1. *Definitions of the gates studied*

Dugan 2004) – that trigger the top fault, as well as its probability to appear. These state-based models are mainly timed Petri nets, in (Adamyant and He 2003), or Markov chains, in (Coppit *et al.* 2000). Deterministic and Stochastic Petri Nets are also used in (Kaiser *et al.* 2007) to analyse quantitatively state/event fault trees, which combine elements of fault tree analysis, Markov chains and state automata. In (Buchacker 2000), an alternative solution to temporal gates is studied with Extended fault trees taking into account repairable components and allowing stochastic dependencies thanks to stochastic Petri nets.

The proposed approach is purely algebraic, as in (Walker and Papadopoulos 2006). It aims at defining the set of laws which is necessary to simplify fault trees with PAND gates upstream, in order to make the state-based models presented above simpler. This paper deals with the trees containing the three gates presented in table 1, whose definitions come from (Vesely *et al.* 1981).

A definition can be commented upon: as stated in (Coppit *et al.* 2000), the term "BEFORE" is ambiguous: it is not clear whether the term is considered strictly – A must appear before B occurs – or not strictly – A must appear before B occurs or at the same time as B occurs.

The hypotheses for this paper are as follows:

- the class of faults studied is limited to non-repairable – persistent – faults.
- there is no restriction on the order of appearance of the faults. Two basic faults can appear simultaneously.
- the term "BEFORE" is considered strictly. This hypothesis is the same as that in (Walker and Papadopoulos 2006).

Taking these hypotheses into account, the expected behaviour of the three gates studied can be represented with timing diagrams. They dis-

play the behaviour of output Q according to the behaviour of inputs A and B. The three cases to consider are as follows: A appears before B appears, A appears at the same time as B appears, A appears after B has appeared.

The expected behaviour of the three gates studied is shown in table 2.

Symbol	Expected behaviour values of A, B and Q at discontinuity points are represented by a black dot
	<p style="text-align: center;">OR Gate</p>
	<p style="text-align: center;">AND Gate</p>
	<p style="text-align: center;">PAND Gate</p>

Table 2. Expected behaviour in the case of non-repairable faults

3. FORMAL FRAMEWORK PROPOSED

The aim of this work is to determine the laws necessary to simplify fault trees containing PAND gates. Consequently the gates of a fault tree were formalized thanks to a mathematical description, to obtain an algebraic form adapted to this kind of calculation. For the sake of clarity during the modelling phase, each concept and element present in a fault tree were mathematically defined. Section 3.1 deals with this. The mathematical model of gates OR and AND, together with the list of the laws which were demonstrated thanks to it, is presented in section 3.2. Section 3.3 deals with operation "BEFORE" defined in order to model gate PAND. The algebraic model of the three gates studied is given in section 3.4.

3.1 Mathematical definition of fault tree concepts

3.1.1. Non-repairable faults In the case of non-repairable or persistent faults, the knowledge of the date of appearance of a fault allowed us to describe the fault without any ambiguity, provided the occurrence of a fault is considered as instant. Before that date, the fault is absent; at that date

and after that date, the fault is present. Taking these hypotheses into account, a non-repairable fault is shown in the timing diagram in figure 2.

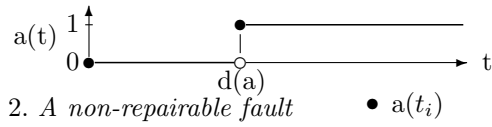


Fig. 2. A non-repairable fault • $a(t_i)$

From a mathematical point of view, a non-repairable fault is a piecewise right-continuous function on $\mathbb{R}^+ \cup \{+\infty\}$, whose range is $\mathbb{B} = \{0,1\}$, and which allows at most a single change of value – a single discontinuity. This function has the following characteristics:

- it is worth 0 as long as there is no fault
- it is worth 1 as soon as the fault has appeared – at the instant noted $d(a)$ on the timing diagram in figure 2

Let \mathcal{F}_{nr} be the set of non-repairable faults. \mathcal{F}_{nr} contains two elements that are special since they are constant. The *always-present fault* is noted e : $\forall t \in \mathbb{R}^+ \cup \{+\infty\}, e(t) = 1$. The *never-occurring fault* is noted ϵ : $\forall t \in \mathbb{R}^+ \cup \{+\infty\}, \epsilon(t) = 0$.

Definition 1. (Date of appearance of a fault). The date of appearance of a fault a is the instant t at which fault a appears. This date is noted $d(a)$.

In regards to the always-present fault and the never-occurring fault, $d(e) = 0$ and $d(\epsilon) = +\infty$.

Definition 2. (Simultaneous faults). Let a and b be two elements of \mathcal{F}_{nr} . Faults a and b are *simultaneous* – noted $a \approx b$ – if and only if they have the same date of appearance: $d(a) = d(b)$.

3.1.2. Fault functions A fault tree describes how basic faults must combine to form the top fault. From a mathematical point of view, the top fault can be considered as the image of the n -tuple of the basic faults by the fault function whose expression is the considered fault tree.

Definition 3. (Fault function). A fault function of order n is an application from $(\mathcal{F}_{nr})^n \rightarrow \mathcal{F}_{nr}$ with $n \in \mathbb{N}^*$.

Let $\Psi_n = \{f : (\mathcal{F}_{nr})^n \rightarrow \mathcal{F}_{nr}\}$ be the set of the fault functions of order n . Ψ_n contains two fault functions that are special because they are constant. These fault functions are noted \top and \perp and are defined as follows. Henceforth, \mathcal{A} is any n -tuple (a_1, a_2, \dots, a_n) of $(\mathcal{F}_{nr})^n$.

Definition 4. (Function \top). \top is the fault function which associates the always-present fault e to any n -tuple $\mathcal{A} \in (\mathcal{F}_{nr})^n$:

$$\begin{aligned} \top : (\mathcal{F}_{nr})^n &\longrightarrow \mathcal{F}_{nr} \\ (a_1, a_2, \dots, a_n) &\longmapsto \epsilon \end{aligned}$$

Definition 5. (Function \perp). \perp is the fault function which associates the never-occurring fault ϵ to any n -tuple $\mathcal{A} \in (\mathcal{F}_{nr})^n$:

$$\begin{aligned} \perp : (\mathcal{F}_{nr})^n &\longrightarrow \mathcal{F}_{nr} \\ (a_1, a_2, \dots, a_n) &\longmapsto \epsilon \end{aligned}$$

Definition 6. (Equivalent fault functions). Let f and g be two elements of Ψ_n . Fault functions f and g are *equivalent* – noted $f \sim g$ – if and only if $\forall \mathcal{A} \in (\mathcal{F}_{nr})^n, f(\mathcal{A}) \approx g(\mathcal{A})$.

Comment: The concepts of *fault* and *fault function* are both mathematically described by functions, but these functions are of different natures:

- *faults* are functions from $\mathbb{R}^+ \cup \{+\infty\} \rightarrow \mathbb{B}$ whose set is noted \mathcal{F}_{nr}
- *fault functions* are functions from $(\mathcal{F}_{nr})^n \rightarrow \mathcal{F}_{nr}$ whose set is noted Ψ_n

3.1.3. Expression of a fault function A fault tree is the expression shown in a graphical form of a fault function defined on the n -tuple of the basic faults present in that tree. The fault tree shown in figure 1 is the graphical representation of the fault function which associates the top fault s to the quadruple of basic faults (a, b, c, d) .

Definition 7. (Equivalent trees). Two fault trees are *equivalent* if and only if they describe the same fault function or two equivalent fault functions.

The simplification of the tree consists of finding an equivalent expression of the fault function free of redundant parts.

The link between a fault tree and a fault function is the same as between a Boolean expression and a Boolean function. This link is as follows:

- a Boolean function can be represented by many Boolean expressions
- two Boolean expressions are equivalent if they represent the same Boolean function
- the link between two equivalent Boolean expressions depends on the laws established for operations defined on the set of Boolean functions (Grimaldi 2003)

By analogy to Boolean expressions and Boolean functions, fault trees have been simplified by using the laws established for the operations defined on the set of fault functions.

3.1.4. Mathematical model of fault tree elements

Each kind of gate of a fault tree is characterized

by an operation on a set of fault functions. It is the set of the fault functions defined on the n -tuple of the basic faults present in the fault tree. The top fault and each intermediate fault are the images of the n basic faults by a fault function. In the interest of homogeneity, each leaf of the tree should also be considered as a fault function. That fault function is special since the image of the n -tuple of basic faults by that function is the basic fault having the same name.

Definition 8. (Basic fault function). The basic fault functions are the n fault functions of order n σ_i – with $i \in \{1, 2, \dots, n\}$ – defined by:

$$\begin{aligned} \sigma_i : (\mathcal{F}_{nr})^n &\longrightarrow \mathcal{F}_{nr} \\ (a_1, a_2, \dots, a_i, \dots, a_n) &\longmapsto a_i \end{aligned}$$

Each leaf of the fault tree presented in figure 1 is one of the four basic fault functions.

3.2 Operations OR and AND

Be $f, g, h \in \Psi_n$.

3.2.1. Definition of operations OR and AND

These operations model the behaviour of gates OR and AND shown in table 2.

Definition 9. (Operation OR).

$$\begin{aligned} + : \Psi_n \times \Psi_n &\longrightarrow \Psi_n \\ (f, g) &\longmapsto f + g \end{aligned}$$

$f + g$ being defined, for all $\mathcal{A} \in (\mathcal{F}_{nr})^n$, by:

$$(f + g)(\mathcal{A}) = \begin{cases} f(\mathcal{A}) & \text{if } d(f(\mathcal{A})) < d(g(\mathcal{A})) \\ g(\mathcal{A}) & \text{if } d(f(\mathcal{A})) > d(g(\mathcal{A})) \\ f(\mathcal{A}) & \text{if } d(f(\mathcal{A})) = d(g(\mathcal{A})) \end{cases}$$

Definition 10. (Operation AND).

$$\begin{aligned} \cdot : \Psi_n \times \Psi_n &\longrightarrow \Psi_n \\ (f, g) &\longmapsto f.g \end{aligned}$$

$f.g$ being defined, for all $\mathcal{A} \in (\mathcal{F}_{nr})^n$, by:

$$(f.g)(\mathcal{A}) = \begin{cases} g(\mathcal{A}) & \text{if } d(f(\mathcal{A})) < d(g(\mathcal{A})) \\ f(\mathcal{A}) & \text{if } d(f(\mathcal{A})) > d(g(\mathcal{A})) \\ f(\mathcal{A}) & \text{if } d(f(\mathcal{A})) = d(g(\mathcal{A})) \end{cases}$$

Comment: These definitions appear to privilege f with regard to g when $d(f(\mathcal{A})) = d(g(\mathcal{A}))$; that is not the case, since operations OR and AND have been demonstrated as being commutative.

3.2.2. Laws of operations OR and AND These two definitions allowed the demonstration of the following 14 laws:

$$f + g \sim g + f \quad (1)$$

$$f.g \sim g.f \quad (2)$$

$$f + (g + h) \sim (f + g) + h \quad (3)$$

$$f.(g.h) \sim (f.g).h \quad (4)$$

$$f + f \sim f \quad (5)$$

$$f.f \sim f \quad (6)$$

$$f + (g.h) \sim (f + g).(f + h) \quad (7)$$

$$f.(g + h) \sim (f.g) + (f.h) \quad (8)$$

$$f + (f.g) \sim f \quad (9)$$

$$f.(f + g) \sim f \quad (10)$$

$$f + \perp \sim f \quad (11)$$

$$f.\top \sim f \quad (12)$$

$$f + \top \sim \top \quad (13)$$

$$f.\perp \sim \perp \quad (14)$$

The obtention of laws 1 to 10 was mandatory in order to guarantee that this formal framework could replace the classical Boolean algebra for the calculation of the minimal cut sets of static fault trees with non-repairable faults.

The main benefit of this formal framework is to allow the definition of a new operation to take into account the sequence order between faults.

3.3 Operation BEFORE

This operation was introduced to model the concept of priority between faults on which the definition of gate PAND depends.

3.3.1. Definition The expected behaviour of the composition of f and g by operation BEFORE – noted $f \triangleleft g$ – is illustrated by the timing diagrams in figure 3 – Case 1: $d(f(\mathcal{A})) < d(g(\mathcal{A}))$, Case 2: $d(f(\mathcal{A})) = d(g(\mathcal{A}))$, Case 3: $d(f(\mathcal{A})) > d(g(\mathcal{A}))$.

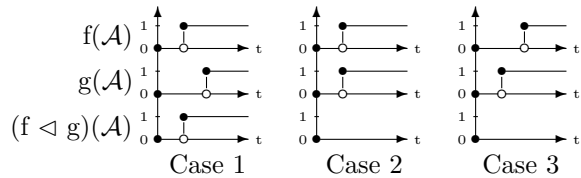


Fig. 3. Expected behaviour for $(f \triangleleft g)(\mathcal{A})$

Definition 11. (Operation BEFORE).

$$\begin{aligned} \triangleleft : \Psi_n \times \Psi_n &\longrightarrow \Psi_n \\ (f, g) &\longmapsto f \triangleleft g \end{aligned}$$

$f \triangleleft g$ being defined, for all $\mathcal{A} \in (\mathcal{F}_{nr})^n$, by:

$$(f \triangleleft g)(\mathcal{A}) = \begin{cases} f(\mathcal{A}) & \text{if } d(f(\mathcal{A})) < d(g(\mathcal{A})) \\ \perp(\mathcal{A}) & \text{if } d(f(\mathcal{A})) \geq d(g(\mathcal{A})) \end{cases}$$

3.3.2. Laws of operation BEFORE The definition of these three operations allowed us to determine the 14 following laws:

$$f + (f \triangleleft g) \sim f \quad (15)$$

$$g + (f \triangleleft g) \sim f + g \quad (16)$$

$$f.(f \triangleleft g) \sim f \triangleleft g \quad (17)$$

$$f + ((f \triangleleft g).h) \sim f \quad (18)$$

$$f \triangleleft (g + h) \sim (f \triangleleft g).(f \triangleleft h) \quad (19)$$

$$(f + g) \triangleleft h \sim (f \triangleleft h) + (g \triangleleft h) \quad (20)$$

$$f \triangleleft (g.h) \sim (f \triangleleft g) + (f \triangleleft h) \quad (21)$$

$$(f.g) \triangleleft h \sim (f \triangleleft h).(g \triangleleft h) \quad (22)$$

$$(f \triangleleft g).(g \triangleleft h).(f \triangleleft h) \sim (f \triangleleft g).(g \triangleleft h) \quad (23)$$

$$(f \triangleleft g).(g \triangleleft f) \sim \perp \quad (24)$$

$$f \triangleleft f \sim \perp \quad (25)$$

$$f \triangleleft \perp \sim f \quad (26)$$

$$\perp \triangleleft f \sim \perp \quad (27)$$

$$f \triangleleft \top \sim \perp \quad (28)$$

The algebraic definition of this operation was one of the objectives of this work. The homogeneous framework defined above thus allows the modelling and algebraic manipulation of fault trees with PAND gates.

3.4 Algebraic model of the gates

Table 3 recalls the elements of table 1 and presents the algebraic model associated with each gate.

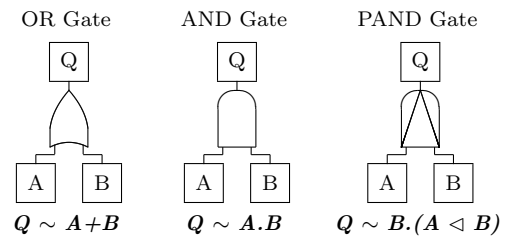


Table 3. Algebraic model of gates OR, AND and PAND

Regarding gate PAND, the following simplified form can be found:

$$\begin{aligned} Q &\sim (A.B).(A \triangleleft B) \stackrel{(2,4)}{\sim} B.(A.(A \triangleleft B)) \\ &\stackrel{(17)}{\sim} B.(A \triangleleft B) \end{aligned}$$

4. SIMPLIFICATION OF FAULT TREES WITH PAND GATES

Determining laws 15 to 28 was the operational objective of this work. Combined with laws 1 to 14, they allow the simplification of the fault trees which contain gate PAND. These laws are

sufficient to analyse any fault tree which does not contain imbricated combinations of PAND gates – i.e. PAND gates whose input sub-trees contain PAND gates.

In order to simplify any tree containing PAND gates whatever its structure, the development laws of forms $f \triangleleft (g \triangleleft h)$ and $(f \triangleleft g) \triangleleft h$ are mandatory.

The simplification of a fault tree consists in:

- (1) expressing the output of each gate according to its inputs, thanks to the algebraic model given in table 3
- (2) developing the expression obtained by applying laws (8)(19)(20)(21)(22) as many times as necessary
- (3) simplifying it by applying the other laws as many times as necessary
- (4) building the fault tree related to the simplified expression obtained

The application of this method to the tree in figure 1 allows us to obtain the equivalent fault tree given in figure 4. Its simplification is as follows:

$$\begin{aligned}
 s &\sim m + n \sim (a + p) + (d.(q \triangleleft d)) \\
 &\sim (a + (a.b)) + (d.((a + c) \triangleleft d)) \\
 &\stackrel{(20)}{\sim} (a + (a.b)) + (d.((a \triangleleft d) + (c \triangleleft d))) \\
 &\stackrel{(3,8)}{\sim} (a + (a.b)) + d.(a \triangleleft d) + d.(c \triangleleft d) \\
 &\stackrel{(9)}{\sim} a + d.(a \triangleleft d) + d.(c \triangleleft d) \\
 &\stackrel{(2,18)}{\sim} a + d.(c \triangleleft d)
 \end{aligned}$$

The simplification technique presented above has been validated by the development of a symbolic calculus module on Mathematica[®] dedicated to the simplification of fault trees with PAND gates. It contains all the laws presented in this paper.

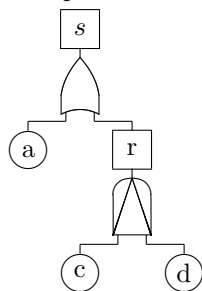


Fig. 4. Fault Tree equivalent to Fault Tree in figure 1

5. CONCLUSION

The first contribution of this work is the definition of a homogeneous and unambiguous framework for the formalization of gate PAND. This framework allowed us to determine the laws necessary to the simplification of fault trees containing PRIORITY AND gates. The hypotheses made were

as follows: basic faults are non-repairable faults, they can appear simultaneously, and the notion of priority is considered strictly.

On-going work deals with determining the developed forms of the two laws presented in section 4, which will allow the simplification of fault trees with PAND gates compositions.

In the short term, the consequences of the hypothesis made on priority – strict priority – will be studied by developing a new operation for which the output fault would also occur if both input faults occur simultaneously, as it was supposed in (Coppit *et al.* 2000). The extension of the work to other temporal gates – such as gate SEQ (Coppit *et al.* 2000) – is also planned.

REFERENCES

- Adamyany, A. and D. He (2003). Sequential failure analysis using counters of petri net models. *IEEE Transactions on Systems, Man, and Cybernetics - part A: Systems and Humans* **33**(1), 1–11.
- Buchacker, K. (2000). Modeling with extended fault trees. *Proceedings of the High Assurance System Engineering Symposium (HASE'2000)* pp. 238–246.
- Coppit, D., K.J. Sullivan and J.B. Dugan (2000). Formal semantics of models for computational engineering: a case study on dynamic fault trees. *International Symposium on Software Reliability Engineering (ISSRE'2000)* pp. 270–282.
- Grimaldi, R.P. (2003). *Discrete and Combinatorial Mathematics*. 5 ed.. Addison-Wesley.
- IEC (2006). IEC standard 61025 (Ed. 2.0): Fault tree analysis.
- Kaiser, B., C. Gramlich and M. Forster (2007). State/event fault trees - a safety analysis model for software-controlled systems. *Reliability Engineering and System Safety*. doi: 10.1016/j.res.2006.10.010.
- Tang, Z. and J.B. Dugan (2004). Minimal cut set/sequence generation for dynamic fault trees. *Annual Reliability and Maintainability Symposium 2004 Proceedings*.
- Vesely, W.E., F.F. Goldberg, N.H. Roberts and D.F. Haasl (1981). *Fault Tree Handbook*. Washington D.C., USA, US Nuclear Regulatory Commission.
- Vesely, W.E., M. Stamatelatos, J.B. Dugan, J. Fragola, J. Minarick III and J. Railsback (2002). *Fault Tree Handbook with Aerospace Applications*. Washington D.C., USA, NASA Office of Safety and Mission Assurance.
- Walker, M. and Y. Papadopoulos (2006). Pandora: The time of priority-and gates. *INCOM'06, 12th IFAC Symposium on Information Control Problems in Manufacturing*.