Modeling and analysis of causes and consequences of failures

Seppo Virtanen, Ph. D., Tampere University of Technology
Per-Erik Hagmark, Ph. D., Tampere University of Technology
Jussi-Pekka Penttinen, M. Sc., Tampere University of Technology

Key Words: cause, consequence, event, logic, modeling, simulation

SUMMARY & CONCLUSIONS

This paper presents a computer-supported method for modeling and analyzing causes and consequences of failures. The developed method is one of the main results from a nine-year research project, which was completed in February 2005 and carried out by Tampere University of Technology.

The applicability of the developed methods and software has been tested in the companies, which have been involved in the research project. The participating companies are both manufacturers and users in metal, energy, process and electronics industries. Their products and systems have to respond to high safety and reliability demands. Most of the participating companies have started to apply the proposed method and software for modeling and analysis of failure logic for their products and systems. The application of the method forces experts to identify all potential component hardware failures, human errors, possible disturbances and deviations in the process, and environmental conditions related to the selected TOP-event. Based on experience, and with the help of the methods, it is possible to find out those problem areas of the design stage, which can delay product development and/or reduce safety and reliability.

1. *INTRODUCTION*

Modeling and analysis of causes and consequences of failures form a foundation for quantitative investigation of the reliability, safety and risks related to a design entity. The general term "entity" or "design entity" can stand for function, system, equipment, mechanism, or any kind of part.

A "cause tree" consists of such (well-defined) causes and interconnected causalities that can lead to the occurrence of a TOP-event. Thus, a cause tree structure forms a basis for a failure logic model of the design entity in question. A "consequence tree" again describes the possible chains of consequences initiated from a TOP-event. A consequence may further cause other consequences, either exclusively or independently. Finally, a combination of cause trees and a consequence tree, illustrated in Figure 1, will be called a "cause-consequence tree". A cause-consequence tree may for example contain several separate chains of events that lead to the same consequence. (Note the chains to consequences 1 and 2 in Figure 1.)

The cause tree model is used to define the occurrence of the TOP-event, from which the consequences to be studied originate. Conditional relations between consequences may also be modeled precisely by using cause trees. The developed method can further describe relations and shared causes between cause and consequence structures. The consequence tree does not offer any additional logical structure, but it makes it possible to model such consequences, which have conditional relations to the cause tree structures. It is also possible to model and analyze several TOP-events simultaneously.

For the analysis of causes and consequences of failures, the root cause probabilities and the gate probabilities are first estimated, and then the modeled failure logic is analyzed through stochastic simulation. The developed method is simple enough to be applicable also for the analysis of very large models. Notwithstanding, it is still capable to produce exact and useful results.

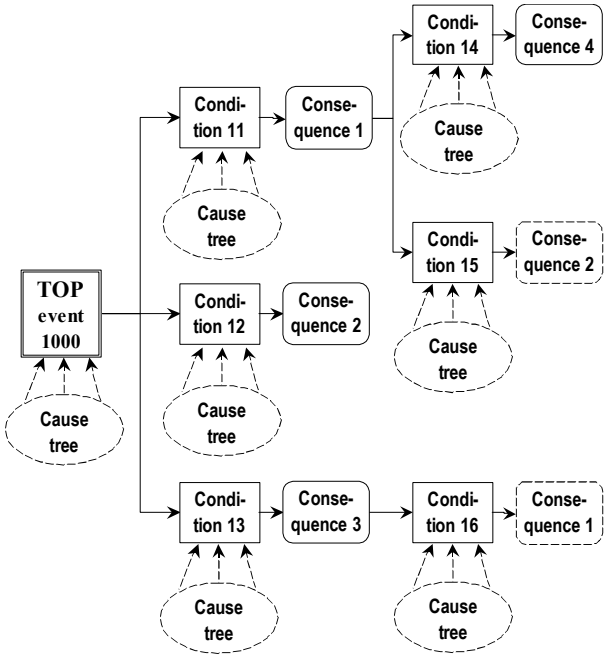

Figure 1. Cause-consequence Structure

The structure of the paper is as follows: In section 2, the developed cause tree model is introduced, and in section 3 the developed method for modeling and analyzing a consequence tree is presented.

2. THE CAUSE TREE MODEL

2.1 *The Gate Model*

An event alone or together with other events can cause a new event. In other words, the *state* of an input event is either $x = 1$ (occurring, true), or $x = 0$ (not occurring, false), and the equivalent state of the *gate* (or *gate event*) is determined with a partly logical and partly stochastic mechanism. The gate mechanism will be characterized by giving the data column

$$(G,\ a,\ b,\ p,\ C_1,\ C_2,\ \ldots,\ C_n)^T \qquad (1)$$

G ID-number of the gate (event), positive integer
a, b Logic parameters, $0 \le a \le b \le n$, integers
p Conditional probability, $0 < p \le 1$
$|\ C_i\ |$ ID-number of input (event), $i = 1, 2, \ldots, n$
$C_i<0$ Input (i) is first negated (NOT operator).

The state of a gate (gate event) G is a random variable depending on the states of the input events:

$$x_G = \Phi\left(a \le \sum_{i=1}^{n} \left| x_{|C_i|} - \Phi(C_i < 0) \right| \le b \right) \cdot \Phi(U \le p) \qquad (2)$$

where U is a random variate from the uniform distribution on the unit interval, and the truth function Φ("statement") equals 1 if "statement" is true, and otherwise 0. Shortly: The logic of the gate is true, if at least a, and at most b of the inputs are true. If so, the gate event is true with probability p.

Example. The gate $(8,\ 1,\ 2,\ 0.9,\ 1,\ -3,\ 4)^T$ in Figure 2 is an example of a non-monotonic (-3) and stochastic (0.9) gate. The variety generation formula (2) takes now the form: $x_8 = \Phi(1 \le x_1 + (1 - x_3) + x_4 \le 2) \cdot \Phi(U \le 0.9)$.

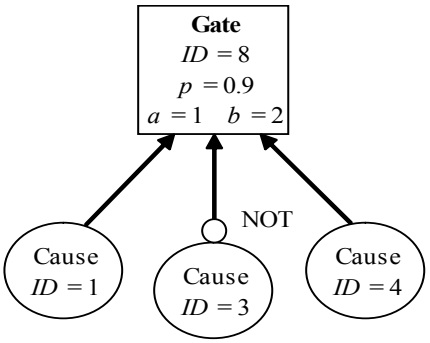

Figure 2. A gate.

By varying the parameters $a, b, p,$ and by using the NOT operator ($C_i<0$), it is possible to model logically complicated and stochastic gates. For example, "inhibit" structure [1] is embedded in our gate model by definition. The probability of the conditional event is just the model parameter $p < 1$ (e.g. gate 8 in fig.2). Further, by choosing $a = b$ in our model we have a natural generalization of the XOR (exclusive-OR) gate. The normal logical XOR gate [2] corresponds of course to the stronger choice $a = b = 1$, $p = 1$. The following table lists the simplest gate types.

Type of gate	***a***	***b***	***p***
OR	1	*n*	1
AND	*n*	*n*	1
Voting, *k*/*n* ($0 \le k \le n$)	*k*	*n*	1
Generalized Inhibit	*a*	*b*	<1
Generalized XOR ($1 \le k < n$)	*k*	*k*	1

2.2 *Cause Tree Structure*

A cause tree is a net of events, where the causally directed connections between the events are assumed not to alter from time to time. The mechanism of the occurrence of a gate event was defined in section 2.1. If an event is not an input to any gate, it is called a *TOP-event*. If an event has no inputs, it is called a *root cause* (or basic event). If wanted, a root cause can also be interpreted as a gate, whose logic is always true: a=0, b=∞.

Example. Different sequences of two causes can be modeled to have different consequences, Figure 3. Observe the Priority-AND structures.

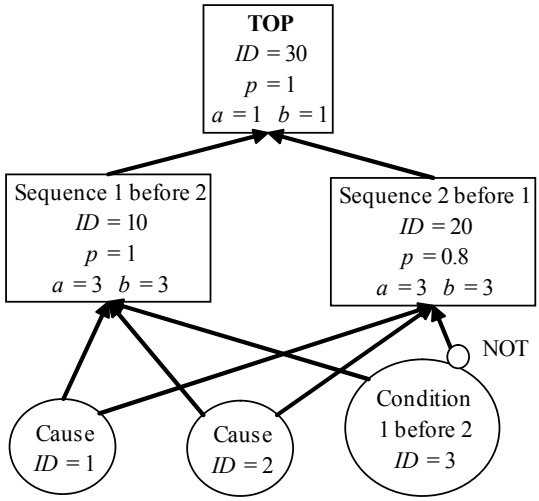

Figure 3. A cause tree with Priority-AND gates

A cause tree can be fully expressed with a *structure matrix* as follows. Each column of the matrix is the data column of a gate (1). Because of the number of gates' inputs various, the shorter ones are filled with zeros to obtain equal lengths for all columns of the matrix. For example, a structure matrix of the cause tree in Figure 3 is

The order of columns is sufficiently determined by the following principle: If a gate is an input to another gate, then its column is located to the left. The simulation of the tree proceeds in the matrix from left to right, so the inputs of a gate are always generated before the gate itself. In other words, the simulation order will not contradict the chronological order.

$$\begin{bmatrix} G & 10 & 20 & 30 \\ a & 3 & 3 & 1 \\ b & 3 & 3 & 1 \\ p & 1 & 0.8 & 1 \\ C_1 & 1 & 1 & 10 \\ C_2 & 2 & 2 & 20 \\ C_3 & 3 & -3 & 0 \end{bmatrix}. \quad (3)$$

2.3 *Construction of cause tree graphic*

During the research project, we have developed the Event Logic Modeling and Analysis Software (ELMAS) tool. After the identification of the events related to the TOP-event, experts examine the generated event list one by one and indicate the event's cause and consequence connections with the other events. Based on the expert's decisions, ELMAS draws the logic diagram on the screen. The same cause can occur in many places in the logic. On the computer screen the expert can drag and drop the events in to the right position based on his/her best understanding of the logic. If the event is moved so that it leads to a loop in the tree, ELMAS gives a warning and rejects the choice. After the causes and consequences of events are determined, the types of gates are defined (Figure 4).

The same cause tree model (2.2) is also used in software for allocation of reliability and availability requirements (RAMalloc), and simulation of reliability and maintenance costs (RAMoptim).

2.4 *Simulation of the failure logic*

If the logic in the modeled cause tree is simple enough, it can be directly studied by using analytical means, for example minimal cut sets. When the causes and their interconnected causalities are complex and the logic is not monotonic, stochastic simulation is highly preferred. The simulation data leads to a variety of useful results.

Before the simulation, the probabilities of the occurrence of the root causes are estimated. For the estimation of these probabilities, several different types of expert judgment methods are integrated into ELMAS. Each of these methods defines the probability within a short period of time, *dt,* or at a random moment.

At the beginning of a simulation process, the random occurrence of the root causes is simulated. After this, the occurrences of the gates are generated in a "chronological" order based on the cause tree matrix. One simulation round defines the complete state of the entire cause tree within a period of length *dt*.

After a sufficient number of simulation rounds, the statistical information about the behavior of the studied object is representative. The developed software assists in defining how many simulation rounds are required. During the simulation, it is possible to see on the screen (a) how many of all the possible combinations have already occurred and (b) an estimation of the maximum probability of the combinations that have not yet occurred.

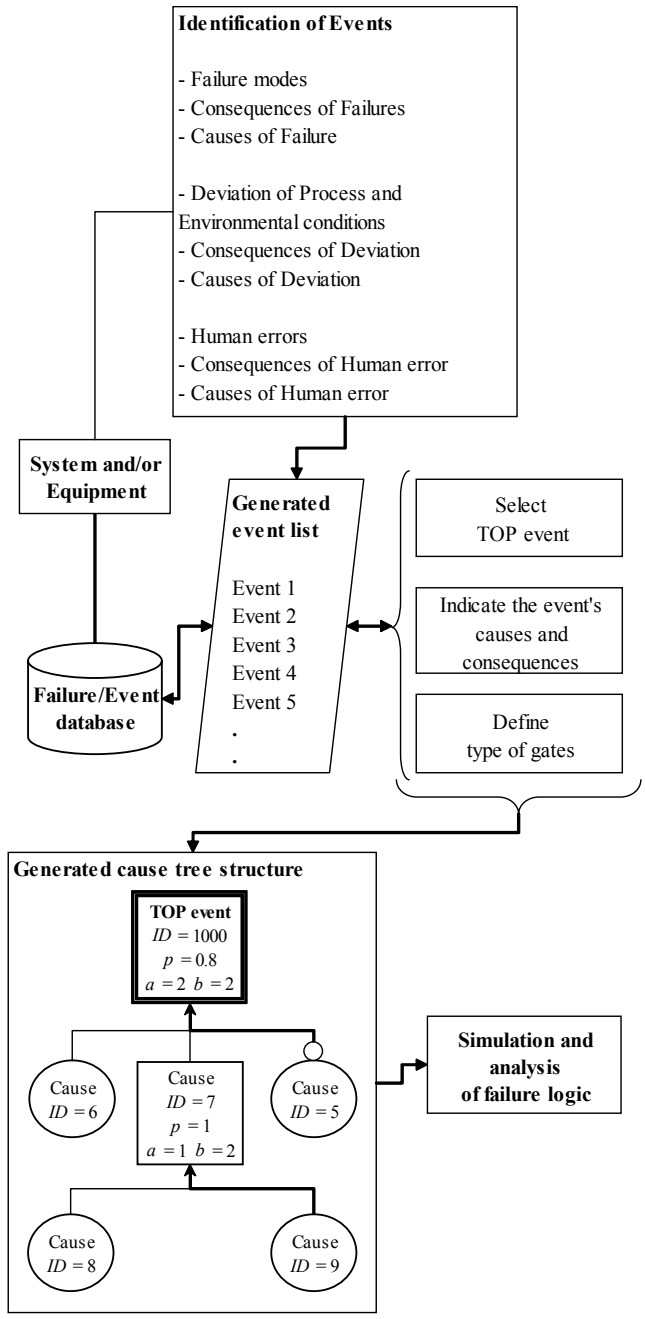

Figure 4. Principle of modeling the cause tree structure related to the selected TOP-event with ELMAS

2.5 *Results of the failure logic analysis*

The raw simulation data can be refined to useful results. The simplest is the (estimated) probability of an event:

$$P_A = \frac{n_A}{n} \quad (4)$$

P_A the probability of event A
n_A the number of occurrences of A
n the total number of simulated rounds.

It is also possible to estimate conditional probabilities. The condition can be the occurrence of an event or a combination

of events:

$$P_{A|X} = \frac{n_{AX}}{n_X} \quad (5)$$

$P_{A|X}$ conditional probability of A under condition X
n_{AX} the number of times A occurred under condition X
n_X the number of occurrences of condition X.

Finally, conditional probabilities can also be calculated for combinations of the states of specific events:

$$P_{C|X} = \frac{n_{CX}}{n_X} \quad (6)$$

$P_{C|X}$ conditional probability of combination C under condition X
n_{CX} the number of times C occurred under condition X
n_X the number of occurrences of condition X.

2.6 *Importance measures*

An importance measure describes correlative relations between two events. The events to be studied may be chosen freely from the model, except that the causality order is required (otherwise, the results would not be meaningful). We consider several importance measures [3]. They all attach slightly different "importance values" to the events, and they may even lead to a different order of importance. Some of them can be found in Figure 5, [4]. All of these importance measures are also integrated into ELMAS.

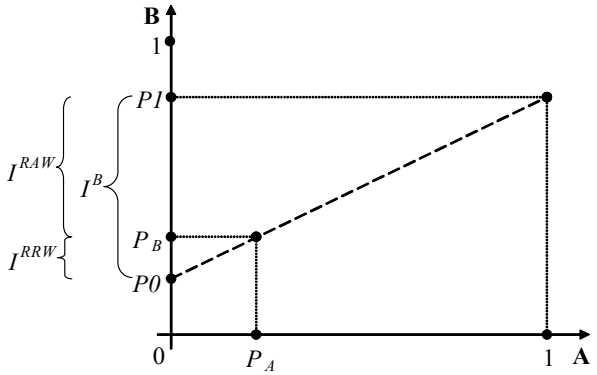

Figure 5. Importance measures

The *importance* of event A (cause) is determined from the event B (consequence) point of view. The current probabilities, P_A and P_B, are assessed from simulation data as described in section 2.5. If P_A will increase to 1, which means that A is always true, P_B will increase to (say) $P1$. Correspondingly, if P_A will reduce to 0, which means B will never occur, P_B will reduce to $P0$.

The probabilities $P0$ and $P1$ are conditional for event B, when event A is occurred or not occurred. Their difference is called the *Birmbaum's importance measure*

$$I^B = P1 - P0 \quad . \quad (7)$$

This figure expresses the scope of how much P_A can affect P_B in general. The *Risk Reduction Worth* describes how much the probability of event B reduces at most, i.e., if the probability of A can be reduced to zero:

$$I^{RRW} = P_B - P0 = I^B \cdot P_A \, . \quad (8)$$

Similarly, the *Risk Achievement Worth* describes how much the probability of B will increase at most, i.e., if the probability of A happens to increase to one (9):

$$I^{RAW} = P1 - P_B = I^B \cdot (1 - P_A) \quad (9)$$

The *Criticality importance* again describes the probability that A is the main cause to the occurrence of B (10).

$$I^{CR} = 1 - \frac{P0}{P_B} = \frac{I^B \cdot P_A}{P_B} = \frac{I^{RRW}}{P_B} \, . \quad (10)$$

2.7 Risk analysis

All events can be given commensurate losses, which represent what will happen if the event occurs. When the losses have been assessed for the selected causes, all needed information for risk analysis is at hand:

$$R_A = P_A \cdot C_A^{\alpha} \quad (11)$$

R_A the risk value of event A
P_A probability of event A
C_A the extent of loss that event A causes.

The parameter α in equation (11) is used to weaken or strengthen the importance of large damages [5]. If $\alpha > 1$, the risks of large damages are amplified. It is also possible to compare the risks of different causes or different combinations of causes by using the conditional probabilities.

3. CAUSE-CONSEQUENCE TREE

The definition of the occurrence of selected TOP-event and the extent of all possible consequences of course forms the basis for performing e.g. a complete risk analysis for the design entity. The *consequence tree* is an additional structure to be used for modeling the consequences of the TOP-event, which itself is modeled with a cause tree. The cause-consequence structure (Figure 1), where conditions between consequences can be modeled with cause trees, makes it possible to create very general models for the propagation of consequences.

The main contribution is here the possibility to define various conditional levels. These levels do not need to be independent. Interconnections can be taken into account. Some causes in the cause tree model can perhaps be defined only after some consequences have occurred. This may also lead to a situation where the occurrence of some consequence determines the occurrences of some root cause in a cause tree.

3.1 *Modeling consequences of events*

The consequence tree consists of events, which have two states like the events in the cause tree model. When a consequence is true (occurring), it may cause other consequences. The propagation of consequences, starting with a TOP-event, proceeds in a way similar to the cause tree. A consequence "gate" can ramify in a pre-defined set of consequences (outputs). Our model offers two branching types, independent or exclusive.

Independent branching means that several alternatives can occur at the same time and with their own probability.

This type is typical for the first level just after TOP. For example, if the TOP-event is a certain type of failure, several types of damage can follow the same failure (human, environmental, property, business, etc).

Exclusive branching means that *at most* one alternative follows. This type is typical for the second level. For example, human damage can be classified as "fatal", "severe" or "minor", and business damage can be classified as "enormous", "big", "average" or "minor".

3.2 *Modeling conditionality between consequence events*

The binary state of a condition between consequences is modeled with a cause tree. This way it is of course possible to describe very complicated conditions. It is also possible to model more conditional levels than only one level before failure and one after. The first conditional level is usually a calendar time level. If the studied object should always operate, this level is trivial. Otherwise it is possible to precisely define the planned non-operational times, for example, maintenance time, by using cause tree structure.

The next level can be e.g. the normal operation level of the studied entity. At this level, all events related to the normal operation are defined. Some of these events are used as conditions in the next levels. The occurrence of these conditions can cause some consequences, which are pre-conditions to the next levels. These pre-conditions may be e.g. different types of failures of the object operation or just situations, where operation of the object is started.

With the developed method, it is thereby possible to model different levels of operation of the studied entity before getting into the failure levels. These operation levels can be, for example, different running modes of the entity. The failures can also be divided into different levels. For example, some failure may cause system overheating, overheating may cause fire, and fire may spread to different parts of the system. The current running mode of the system may affect how the fire will spread. In other words, there could also be connections between an operating level and a failure level.

3.3 *Consequence matrix*

A consequence tree can be described precisely with the corresponding matrix. This consequence matrix is created similarly as the cause tree matrix. Every column of the matrix describes the consequences of a certain consequence event, and the columns are arranged in a chronological order, the simulation order. Every consequence has its own identity code (ID), and the corresponding column contains information about the next level consequences:

$$\left(ID, excl, (cond_1 \mid conseq_1), \ldots, (cond_n \mid conseq_n)\right)^T \quad (12)$$

The field *excl* in the column (12) defines whether the subsequent consequences are exclusive or independent (section 3.1). The field ($cond_i \mid conseq_i$) contains two values. The first one is the ID of the condition of the next level consequence, that is, the ID of some cause from the cause tree structure. The latter value is the ID of the possible next level consequence.

An example of a consequence tree is shown in Figure 6. The corresponding consequence matrix and the cause structure matrix are given by (13) and (14), respectively. The first consequence ID=100 has the second level consequences ID=200 and ID=300, which are exclusive.

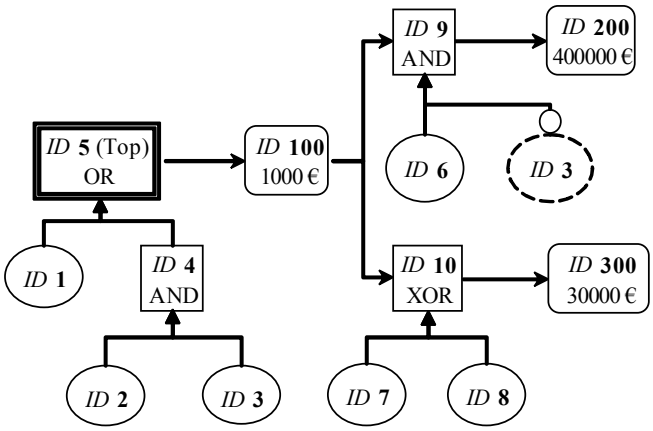

Figure 6. A cause-consequence tree

$$\begin{bmatrix} ID & 0 & 100 \\ exlusive & FALSE & TRUE \\ cond_1 \mid conseq_1 & 5 \mid 100 & 9 \mid 200 \\ cond_2 \mid conseq_2 & & 10 \mid 300 \end{bmatrix} \quad (13)$$

$$\begin{bmatrix} G & 4 & 5 & 9 & 10 \\ a & 2 & 1 & 2 & 1 \\ b & 2 & 2 & 2 & 1 \\ p & 1 & 1 & 1 & 1 \\ C_1 & 2 & 1 & 6 & 7 \\ C_2 & 3 & 4 & -3 & 8 \end{bmatrix} \quad (14)$$

3.4 *Simulation and analysis of cause-consequence tree*

Exactly the same simulation method that is used in the simulation of cause trees is applied to simulate the state of conditions between consequences. The simulation results are also identical with the results of the cause tree simulation. The only difference is that a level of the consequence structure is simulated only if its pre-condition is true.

By the calculation of a consequence level is meant the handling of a column in the consequence matrix. During a simulation round all conditional levels are treated in chronological (i.e., matrix column) order. At least one level is always handled (matrix column ID=0), because the first consequence of the tree is always true. The next level is handled if the consequence of that level has occurred.

In example Figure 6, the causes under gate 5 are at the first level. The causes under gates 9 and 10 are at the second level, which is handled only if consequence 100 is true. Note that root cause 3 belongs to the first level and it is used in the second level, too, as an input to gate 9. There it promotes the occurrence of consequence 200 if it is in the "not occurring" state. After all levels have been handled, one simulation

round is finished.

The example (Figure 6) was analyzed by using the following root cause probabilities:

ID	1	2	3	6	7	8
Prob.	0.0001	0.02	0.03	0.005	0.002	0.04

The simulation of consequence structure consists of only simple testing. Therefore, it does not have remarkable effect on the calculation time. The results from 100000000 simulation rounds were the following:

ID	Loss	Probability	Risk
100	1000	7.03E-4 ± 6.0E-6	~ 0.703
200	400000	4.6E-7 ± 1.4E-7	~ 0.884
300	30000	2.95E-5 ± 1.1E-6	~ 0.184

4. CONCLUSION

The cause-consequence tree method presented above makes it possible to explain and describe precisely the relations between causes and consequences of failures. The causes can be ranked from the probability and/or risk point of view. Results from the analysis help researchers to identify both the most probable causes and chains of causes leading to the TOP-event, and the most significant consequences and chains of consequences. After ranking the causes, a more detailed root cause analysis can be performed by applying the event-cause-consequence method FMEA, which is integrated into ELMAS.

5. REFERENCES

1. Fault Tree Handbook. U.S. Nuclear Regulatory Commission, Nureg-0492. (Co-author David Haasl) ISBN/ISSN 1051-H-02. 1981, p 209.
2. E. J. Henley, H. Kumamoto, *Probabilistic Risk Assessment and Management for Engineers and Scientists*, Second Edition, IEEE Press Piscataway, NJ. 1996, p 597.
3. M.Cheok, G. Parry, R. Serry, "Use of Importance Measures in Risk-Informed Regulatory Applications". *Reliability Engineering and System Safety*, 1998.
4. J. P. Penttinen. *Analysis of failure logic using simulation*, Master' Thesis, Tampere University of Technology. 2005, p 94.
5. J. Norman, McCormick, *Reliability and Risk Analysis, Methods and Nuclear Power Applications.* Academic Press. Inc. 1981, p 446.

BIOGRAPHIES

Seppo Virtanen
Machine Design and Operation Laboratory
Tampere University of Technology
Korkeakoulunkatu 6
FI-33101 Tampere, Finland

e-mail: seppo.virtanen@tut.fi

Seppo Virtanen received his B.Sc., M.Sc. and PhD. degrees from Helsinki University of Technology, Finland. He is currently a Professor in the Machine Design and Operation Laboratory at the Tampere University of Technology. His research and teaching interest includes reliability and maintainability engineering and risk management within a product and system design process. Professor Virtanen has over 15 year's industry experience in the field of reliability engineering and maintenance, which includes three years in energy, pulp and paper industry in USA and two years offshore industry in Norway.

Per-Erik Hagmark, PhD
Machine Design and Operation Laboratory
Tampere University of Technology
Korkeakoulunkatu 6
FI-33101 Tampere, Finland

e-mail: per-erik.hagmark@tut.fi

Per-Erik Hagmark serves on the Machine Design and Operation Laboratory at Tampere University of Technology. He earned his doctoral degree in Mathematics, Applied Mathematics and Theoretical Physics at Helsinki University of Technology in 1983 with a dissertation on generalizations of Walsh functions and fast algorithms. His recent research activities have been around statistics, reliability theory, simulation, and programming.

Jussi-Pekka Penttinen, M.Sc.
Machine Design and Operation Laboratory
Tampere University of Technology
Korkeakoulunkatu 6
FI-33101 Tampere, Finland

e-mail: jussi-pekka.penttinen@tut.fi

Jussi-Pekka Penttinen received his M.Sc. decree in discrete mathematics and software science at Tampere University of Technology in 2005. Last one and half years Mr. Penttinen has worked as a researcher and post-graduate student in the Laboratory of Machine Design and Operation, where he also finished his Master Thesis connected with the analysis of failure logic using simulation.