

## Dependency Modelling Using Fault Tree Analysis

J.D. Andrews, PhD; Department of Mathematical Sciences, Loughborough University, England

J.B. Dugan, PhD; Department of Electrical Engineering, University of Virginia, USA

Keywords; fault tree analysis, dependency, reliability, availability, Markov methods, binary decision diagrams

### Abstract

This paper describes the use of the fault tree method to model the failure probability of systems that feature dependencies. Our method is presented by illustrating its application to an example safety system taken from the offshore industry. Despite the dependencies the representation of the system failure logic retains the fault tree structure by utilising a new gate set. The analysis of the dependent parts of the system is performed by software in a manner, which is totally transparent to the analyst. Markov methods are employed to solve the dynamic, dependent sections of the fault tree and Binary Decision Diagrams to solve the static fault tree sections. It may be necessary to alternate between the two analysis methods several times to solve the complete fault tree structure.

### Introduction

Over the last two decades, Fault Tree Analysis has become an established tool used to assess the likelihood of failure of industrial systems. It is particularly well utilised for the assessment of safety systems whose failure can cause fatalities or have excessive financial penalties. The popularity of the method is due to the ease in which the system failure causality is represented in a logic tree diagram which increases in its resolution as the diagram develops until terminal branch events which represent component failures, software errors or human actions are encountered. This form of diagram, whilst representing a mathematical logic equation, provides a concise, documented means of representing the fault propagation through the engineering system. There are therefore advantages in retaining the basic fault tree structure to develop causes of system failure, whilst extending its analytical capabilities.

Usually the analysis is performed using one of the many commercially available computer software packages. The method used in the packages to perform the calculations is

commonly based on the Kinetic Tree Theory of Vesely [1]. Kinetic Tree Theory was developed in the early 1970's at a time when the pioneering work in systems reliability was being performed. One of the main assumptions of this modelling technique is that the basic events occur independently. In the last decade there has been a significant increase in the complexity of high technology system designs. A feature of many of these modern systems is that they are software controlled and the failure events may exhibit some form of dependency. For the analysis techniques to remain relevant to the modern systems their development must keep pace with those made in the systems technology. Several recent publications have appeared which deal with the research performed in extending the traditional fault tree analysis method to incorporate dependencies [2],[3],[4],[5].

The approach adopted to incorporate dependency modelling has retained the fault tree structure and introduced a new gate set which will enable the dependent sections of the tree structure to be identified and the exact nature of the dependency to be specified. Such gates are described in the sections below. The analysis of such a tree would then be performed by transforming the dependent sections of the fault tree to equivalent Markov state transition diagrams. Construction and analysis of the Markov diagrams is then performed by the analysis software in a manner which is totally transparent to the analyst. Once analysed to produce either the probability or frequency of the intermediate level fault tree events, the results of the Markov assessments would be incorporated back into the original fault tree structure. This process would be continued until a static, independent fault tree structure remains which could then be analysed by traditional fault tree techniques or the more recent Binary Decision Diagram (BDD) methods[6],[7],[8]. Dependent upon the system structure it may at times be necessary to conduct the analysis in an alternating BDD / Markov process to achieve the final results. An example of such a system, a water deluge system on an

offshore platform, is presented in this paper along with a description of its analysis.

Safety System Description (Water Deluge System)

The water deluge system used to illustrate the dependency methodology is shown in figure 1. Whilst this particular system is an example taken from the offshore industry its features are typical of water spray systems used in many different onshore industries. Four pumps are used to provide the water demand to the ringmain. The ringmain transports the water around the platform to the take-off points where it is used to protect against the hazards posed by hydrocarbon fires and explosions. Pressure in the ringmain is

maintained by a jockey pump (not shown in the figure). When the take-off valves open and water is delivered to the spray nozzles the ringmain pressure will drop. Ringmain pressure is monitored and transmitted to the computer control system by the three pressure transmitters (PS1-PS3). When two of the three transmitters indicate a low ringmain pressure the main pumps are activated in the order indicated from top to bottom of the diagram (ie. EP1, EP2, DP1, DP2). As long as two pumps are available then water can be delivered at the required rate to satisfy demand. Four pumps provide redundancy in the system. Pumps 1 and 2 are electric powered and pumps 3 and 4 are the diesel backups.

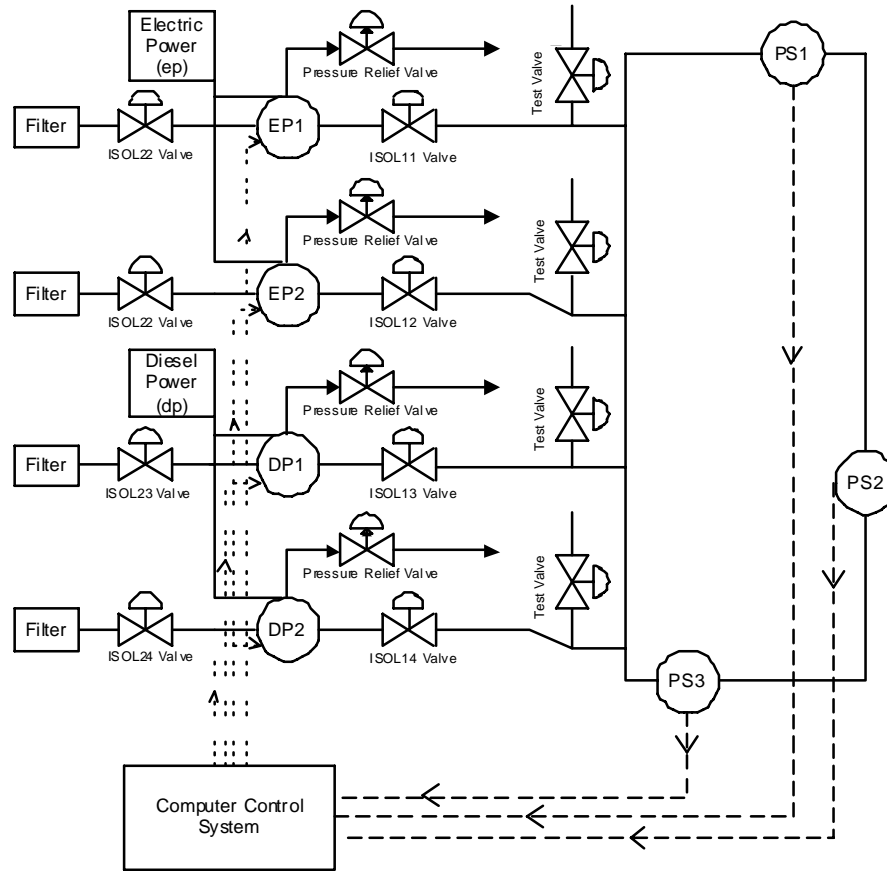


Figure 1 - Schematic representation of the deluge system pump streams

The features on each pump stream are identical. As the water supply is direct from the sea a filter is fitted on each stream. Manual isolation valves are located for maintenance purposes located either side of the pump. A pressure relief valve provides protection for the pump and a test valve

on each line enables individual pumps to be tested without fully activating the deluge system.

There are two failure modes of concern for each stream, the first is that it fails to start (unavailable) and the second is that it fails once

running (unreliable). If a pump stream activates on demand it means that the filter, isolation valves, test valve and pressure relief valve which are all (for this function) passive components are in the working condition. As they are passive they are unlikely to fail in the relatively short running times if they work initially. These are static failure modes. The pump is however is a dynamic component and can also fail once running.

System failure will occur if fewer than two of the four streams can be activated (ie 3 from 4 fail) for the required duration (12 hours)

#### Fault tree model of example safety system

Let us consider the two parts of the system separately when building the fault tree model. That is, we will first consider the computer control system and then consider the pump system. As we analyse this system, we will describe the dependencies that must be modelled, and describe special gates which incorporate these dependencies into the fault tree analysis.

The computer control system consists of the three pressure sensors (of which 2 are needed), plus the hardware and the software. The hardware consists of redundant processors in hot standby mode, each equipped with identical software. While the spare processor is in spare mode, it is monitoring the inputs and outputs of the primary, in order to provide detection and recovery in case of error. When an error is detected, control is switched to the backup processor. The computer control system can thus tolerate a single (detected) hardware or software failure. However, an undetected error causes failure of the computer subsystem regardless of the state of the backup. This latter case (undetected error) is an example of an uncovered fault, which leads to immediate system failure. Another example of an uncovered fault is a software fault which affects both processors simultaneously. One might expect, since the software on both processors is identical, that all software faults would affect both processors. However, there is field data to support the assumption that a large percentage of software faults will affect only a single processor[10]. Modeling uncovered faults is crucial to the analysis of a fault tolerant computer system, and is discussed in more detail in [7] and [9]. A

fault tree model showing the failure of the computer system is shown in figure 2, in which the basic events represent hardware (processors), software and the sensor set.

Each basic event is characterized by information giving the probability of failure (either as a distribution or as a fixed probability) and the probability that a given fault is covered (or uncovered).

Next consider the pump system, consisting of the four pumps, their power sources (two are electric and two are diesel) and their pump streams (associated valves and filters). For now,

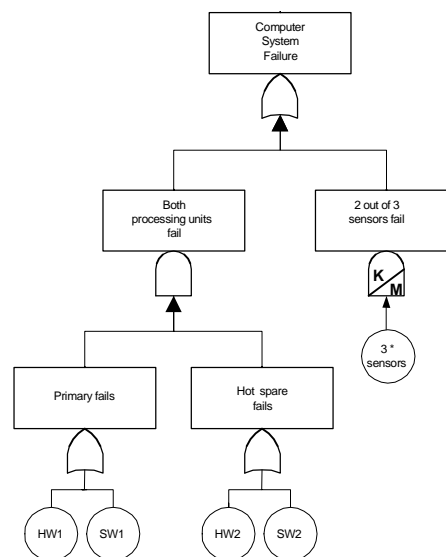


Figure 2 Computer system failure fault tree

let us ignore the pump streams and power supplies, and concentrate on the four pumps.

The set of four pumps operate in standby redundancy in that the two electric pumps are started first, and the diesel pumps provide replacements when the electric pumps are unavailable. On demand, pumps EP1 and EP2 are turned on. If one of these two should fail, it is replaced by DP1. The second pump failure is replaced by pump DP2. This dynamic redundancy scheme introduces dependencies between the failures and requires special modeling techniques. A pump which is in use experiences a different failure rate than one in standby. Therefore, we need to keep track of which pumps are being used and which are in standby. We use a spare gate to model the

failure dependencies which arise from the use of spares.

A spare gate is one of several dynamic gates introduced in [9] and it is used to model several dependencies associated with the use of spares. First, a component which is used as a spare has an associated *dormancy factor* (between zero and one inclusive) which is a multiplicative factor of the active failure rate to produce the spare failure rate. If the dormancy factor is zero, the spare is said to be a cold spare; a cold spare cannot fail before being switched into active operation (failure to activate is modeled as an uncovered failure). If the dormancy factor is unity, then the spare is said to be a hot spare and can fail at the same rate as when active. The in between situation is referred to as a warm spare; a warm spare can fail before switched into active operation, but does so at a lower rate than when active.

The second dependency handled by the spare gate is the use of *pooled* spares, which are spares that can be used as a replacement for whichever of a set of components fails first. Modeling pooled spares requires us to keep track of not only the state of each component, but also the order in which they have failed, so that we can determine which spare is being used where. Further, it might be the case that components have preferences for replacements, in that there is an priority or order in which spares are utilized. This order may well be different for different components.

The spare gate has a set of at least two inputs, the first (leftmost) of which is the designated primary, and the second and subsequent (from left to right) are the spares. When the primary fails, it is replaced (in order) by the spares which are still available (i.e. not failed and not used elsewhere). The single output of the spare gate

returns true when the primary and the spares have been exhausted. Basic events representing spares have failure rates, coverage factors and dormancy factors.

Continuing to ignore the power supplies and pump streams, the fault tree in figure 3 models the pumps and their spares. The pump system fails when there are no longer two available pumps (thus the OR gate with two inputs). The basic events represent the two electric pumps, which are both initially active (on demand). The two diesel pumps (DP1 and DP2) are pooled spares shared by both electric pumps. The first electric pump failure is replaced by DP1 and the second by DP2. Note that if EP2 preferred to be replaced by DP2 then we could switch order the DP1 and DP2 inputs on the second spare gate.

Next let us consider the power supplies. There is an electrical power supply for pumps EP1 and EP2 and a diesel supply for DP1 and DP2. If a power supply fails, then the associated pumps are unavailable (essentially failed). This type of functional dependency of one component on another is easily modeled with a *functional dependency* gate [11]. The functional dependency gate has a trigger input and one or more dependent inputs; when the event associated with the trigger input occurs, the dependent inputs are then forced to occur. The functional dependency gate can be used to model the functional dependence of the pumps on the power supplies : the power supply is the trigger event and the two pumps are the dependent events. The fault tree in figure 4 adds the functional dependency to the fault tree in figure 3. The functional dependency gate produces no output other than the propagation of failures. For this reason it is connected via a dashed line to the rest of the fault tree.

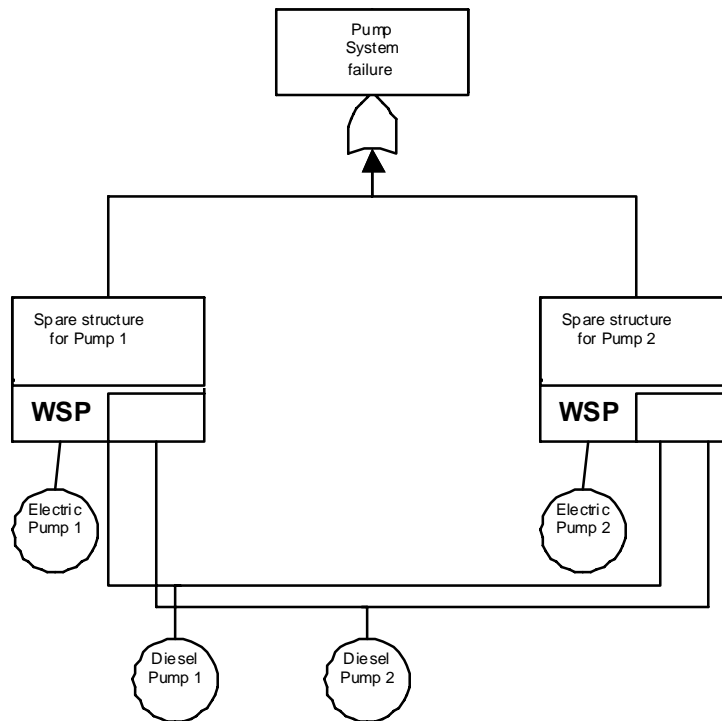


Figure 3 Pump system fault tree structure

An interesting aspect of the model is the inclusion of the pump streams (by which we mean the isolation valves, the pressure relief valves, the test valves and the filter). The pump streams provide support for the pumps and need to be operational in order for the pump to be utilized. We have used two different approaches to including the pump streams, and will describe these approaches in turn. First, we can use the functional dependency gate to model the dependence of each pump on the associated pump stream. Figure 5 shows a fault tree model for a stream, and the functional dependency of the pump on the pump stream. In the full fault tree model, there are then 4 such constructs, one for each pump and stream configuration.

The advantage of this approach is that it is simple to describe and can be solved using Galileo (a software tool for dynamic fault tree analysis) [12], but the disadvantage is that a large Markov

chain is needed for analysis. The Markov chain which is used to solve this system must account for not only the pumps and power supplies, but also for every valve and filter in each channel. Since the number of states in a Markov grows exponentially with the number of components being considered, the resulting model can be quite large. Further, since the pump streams are unlikely to fail once the pump is running, it is not necessary to model each filter and valve in such detail. It is sufficient to know whether the stream is operational on demand.

In [2], an approach is developed for separating the static analysis of the pump stream from the dynamic analysis of the pumps themselves. The probability that the stream is available on demand is determined for each stream, and these probabilities are used to determine the initial state probabilities for the Markov analysis of the pumps and power supplies.

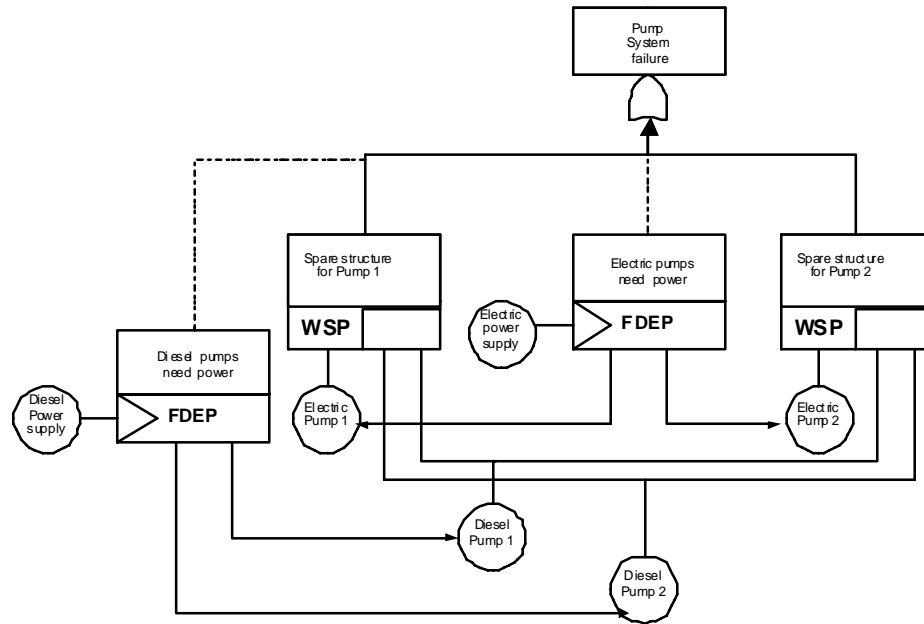


Figure 4 Detailed Pump system fault tree

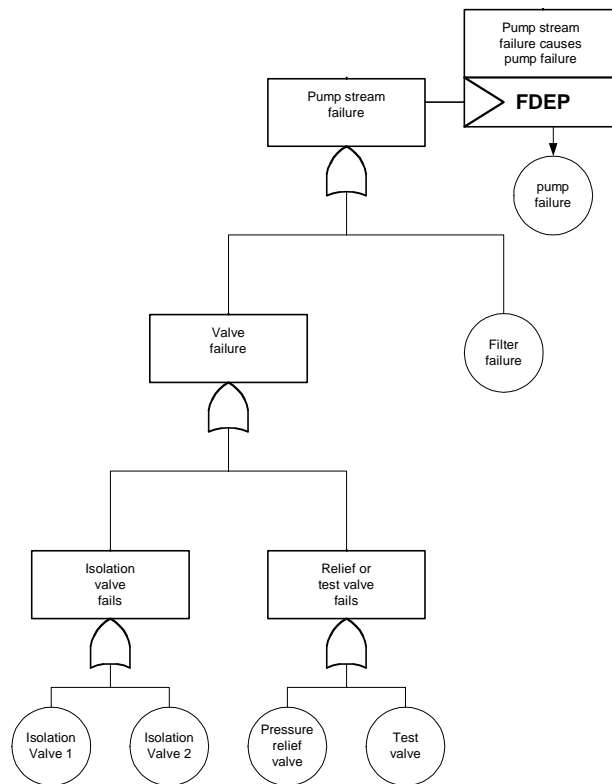


Figure 5 Pump Stream Fault Tree

Analysis Process

The final fault tree for the failure of the pumping system has a top event whose cause is given by the failure of the computer system (figure 2) OR

the failure of the pumping system (Figures 4 and 5 – the functional dependencies modelled in the figure 5 fault tree feed into the four pump failure events in figure 4). The procedure for analysis first involves identifying the modules of the tree

which have dependencies and those which are independent. A bottom up analysis scheme is then implemented where the lowest level modules are analysed using the appropriate technique (BDD's for independent sections and Markov models for dependencies) and the results fed into the analysis of the higher level sections. This is performed as an alternating sequence of BDD and Markov models until the results for the top event are obtained.

The lowest level modules in the pump system fault tree are the functional dependency gates represented in figure 5. All events contributing to the cause of the functional dependency event are independent and therefore these sections can be analysed using the BDD technique. This will produce the pump stream failure probabilities  $P_{E1}$ ,  $P_{E2}$ ,  $P_{D1}$ , and  $P_{D2}$ .

Proceeding up the tree structure the next section for analysis is the Pump system section represented by the fault tree in figure 4. This has

basic events with functional dependencies and will therefore require a Markov assessment. The Markov model has been listed as a table (Table 1) for clarity. The first column represents the state number, the remaining columns identify the status of each pump, coded as W for working, S for standby and F for failed. The status thus indicated is at the start of the pumping process. The failure status for a pump is therefore caused by either the pump itself failing prior to the demand whilst dormant, the associated pump stream failure (the probability determined by the first level BDD analysis) or the power source unavailability. Standby status indicates that the pump is functional but not operating when the demand occurs. It can then fail following the demand due to a dynamic failure event. The pumping process is required for 12 hours to mitigate the hazard. It is assumed that in this short period of time repair action can not be completed and that passive components (valves, pipework etc) cannot fail.

STATE NO.	STREAM STATUS				INITIAL PROBABILITY
	1	2	3	4	
1	W	W	S	S	$(1-P_{E1})(1-P_{E2})(1-P_{D1})(1-P_{D2})$
2	F	W	W	S	$P_{E1}(1-P_{E2})(1-P_{D1})(1-P_{D2})$
3	W	F	W	S	$(1-P_{E1})P_{E2}(1-P_{D1})(1-P_{D2})$
4	W	W	F	S	$(1-P_{E1})(1-P_{E2})P_{D1}(1-P_{D2})$
5	W	W	S	F	$(1-P_{E1})(1-P_{E2})(1-P_{D1})P_{D2}$
6	F	F	W	W	$P_{E1}P_{E2}(1-P_{D1})(1-P_{D2})$
7	F	W	F	W	$P_{E1}(1-P_{E2})P_{D1}(1-P_{D2})$
8	F	W	W	F	$P_{E1}(1-P_{E2})(1-P_{D1})P_{D2}$
9	W	F	F	W	$(1-P_{E1})P_{E2}P_{D1}(1-P_{D2})$
10	W	F	W	F	$(1-P_{E1})P_{E2}(1-P_{D1})P_{D2}$
11	W	W	F	F	$(1-P_{E1})(1-P_{E2})P_{D1}P_{D2}$
12	F	F	F	W	$P_{E1}P_{E2}P_{D1}(1-P_{D2})$
13	F	F	W	F	$P_{E1}P_{E2}(1-P_{D1})P_{D2}$
14	F	W	F	F	$P_{E1}(1-P_{E2})P_{D1}P_{D2}$
15	W	F	F	F	$(1-P_{E1})P_{E2}P_{D1}P_{D2}$
16	F	F	F	F	$P_{E1}P_{E2}P_{D1}P_{D2}$

Table 1 Markov Diagram State List

Initial probabilities of entering each of the states in the table is determined using the results of the pump stream dormant failure fault trees. The transition rates between the states are then obtained due to failure of the pumps, listed in Table 2 and their power supply failures, listed in Table 3. The two sets of transition rates are superimposed onto the state transition diagram

for analysis. System failure states are those with less than two functional pumps i.e. 12-16. Performing the analysis on the Markov diagram then yields the failure probability for the pump system. Progressing up to the top level in the fault tree structure will combine this probability with that of the computer system failure to obtain the overall system unavailability.

From state	To state	Transition rate	From State	To state	Transition rate
1	2	$\lambda_{e1}$	8	13	$\lambda_{e2}$
1	3	$\lambda_{e2}$	8	14	$\lambda_{d1}$
2	6	$\lambda_{e2}$	9	12	$\lambda_{e1}$
2	7	$\lambda_{d1}$	9	15	$\lambda_{d2}$
3	6	$\lambda_{e1}$	10	13	$\lambda_{e1}$
3	9	$\lambda_{d1}$	10	15	$\lambda_{d1}$
4	7	$\lambda_{e1}$	11	14	$\lambda_{e1}$
4	9	$\lambda_{e2}$	11	15	$\lambda_{e2}$
5	8	$\lambda_{e1}$	12	16	$\lambda_{d2}$
5	10	$\lambda_{e2}$	13	16	$\lambda_{d1}$
6	12	$\lambda_{d1}$	14	16	$\lambda_{e2}$
6	13	$\lambda_{d2}$	15	16	$\lambda_{e1}$
7	12	$\lambda_{e2}$	7	14	$\lambda_{d2}$

Table 2 pump failure transition rates

From state	To state	Transition rate	From state	To state	Transition rate
1	6	$\lambda_{ep}$	1	16	$\lambda_{dp}$
2	6	$\lambda_{ep}$	2	14	$\lambda_{dp}$
3	6	$\lambda_{ep}$	3	15	$\lambda_{dp}$
4	12	$\lambda_{ep}$	4	11	$\lambda_{dp}$
5	13	$\lambda_{ep}$	5	11	$\lambda_{dp}$
7	12	$\lambda_{ep}$	6	16	$\lambda_{dp}$
8	13	$\lambda_{ep}$	7	14	$\lambda_{dp}$
9	12	$\lambda_{ep}$	8	14	$\lambda_{dp}$
10	13	$\lambda_{ep}$	9	15	$\lambda_{dp}$
11	16	$\lambda_{ep}$	10	15	$\lambda_{dp}$
14	16	$\lambda_{ep}$	12	16	$\lambda_{dp}$
15	16	$\lambda_{ep}$	13	16	$\lambda_{dp}$

Table 3 Power failure transition rates

Summary and Conclusions

We have described, by means of a representative example, a methodology for incorporating the

analysis of various kinds of dependencies into a fault tree. The dependencies considered include functional dependencies, static (on-demand) dependencies, sharing relationships and uncovered faults. These dependencies arise naturally in mechanical, electrical and computer based systems, and their correct analysis is crucial to the accurate assessment of the system reliability.

### References

1. Veseley, W.E., "A time dependent methodology for fault tree evaluation", Nucl. Eng Des., 13, 337-360, 1970.
2. Ridley L.M. and Andrews J.D., "Optimal design of systems with standby dependencies", to be published in Quality and Reliability Engineering International, 15, 1999.
3. Andrews J.D. and Ridley L.M., "Analysis of systems with standby dependencies", Proceedings of the International System Safety Conference, Seattle, Sept 1998.
4. Rohit Gulati and Joanne Bechta Dugan, "A modular approach for analyzing static and dynamic fault trees," in *Proceedings of the Reliability and Maintainability Symposium*, January 1997
5. Ragavan Manian, David Coppit, Kevin J. Sullivan and Joanne Bechta Dugan, "Bridging the gap between systems and dynamic fault tree models," *Proceedings of the 1999 Reliability and Maintainability Symposium*, January 1999, pages 105-111.
6. Rauzy A., "A brief introduction to Binary Decision Diagrams", Eur. J. Automat., 30(8), 1996.
7. Dugan J.B. and Doyle S.A., "Incorporating imperfect coverage into binary decision diagrams", Eur. J. Automat., 30(8), 1996.
8. Sinnamon R.M. and Andrews J.D., "Quantitative fault tree analysis using binary decision diagrams, Eur. J. Automat., 30(8), 1996.
9. Joanne Bechta Dugan, Salvatore Bavuso, and Mark Boyd, "Fault trees and Markov models for reliability analysis of fault tolerant systems," *Reliability Engineering and System Safety*, 39:291-307, 1993.
10. I. Lee and R.K. Iyer, "Faults, Symptoms and Software Fault Tolerance in the

Tandem GUARDIAN90 Operating System," *Proceedings of the 23<sup>rd</sup> International Symposium on Fault Tolerant Computing*, June 1993.

11. Joanne Bechta Dugan, Salvatore J. Bavuso and Mark A. Boyd, "Dynamic fault tree models for fault tolerant computer systems," *IEEE Transactions on Reliability*, Volume 41, Number 3, pages 363-377, September 1992.
12. Kevin J. Sullivan, David Coppit and Joanne Bechta Dugan, "The Galileo Fault Tree Analysis Tool," *Proceedings of the 1999 Fault Tolerant Computing Symposium (FTCS-29)*, June 1999. (Also see the web page [www.cs.virginia.edu](http://www.cs.virginia.edu)).

### Biography

John D. Andrews, PhD, Department of Mathematical Sciences, Loughborough University, Loughborough, LE11 3TU, England. e-mail [J.D.Andrews@lboro.ac.uk](mailto:J.D.Andrews@lboro.ac.uk)

Dr Andrews is a Senior Lecturer in the Department of Mathematical Sciences at Loughborough University. He joined this department in 1989 having previously gained nine years industrial experience at British Gas and two years lecturing experience at the University of Central England.

His current research interests concern the assessment of the safety and risks of potentially hazardous industrial systems. This research has been heavily supported by funding from industry. Recent grants have been secured from Mobil North Sea Ltd, Daimler Chrysler and Rolls Royce Aero Engines.

Joanne Bechta Dugan, PhD, Department of Electrical Engineering, University of Virginia, Thornton Hall, Charlottesville, VA 22903-2442 USA. e-mail [jbd@Virginia.edu](mailto:jbd@Virginia.edu)

Joanne Bechta Dugan is a Professor of Electrical Engineering at the University of Virginia, and was previously Associate Professor of Computer Science at Duke University and visiting Scientist at the Research Triangle Institute. She has performed and directed research on the development and application of techniques for the analysis of computer systems which are

designed to tolerate hardware and software faults. Dr. Dugan is Senior Associate Editor of the *IEEE Transactions on Reliability*, is a Senior member of the IEEE (Reliability and Computer Societies). She served on the National Research Council Committee on Application of Digital Instrumentation and Control Systems to Nuclear Power Plant Operations and Safety.

#### Acknowledgement

The authors would like to acknowledge the financial support of NATO which has enabled the collaboration between Loughborough University, England and University of Virginia, USA in developing methods to predict the reliability of safety critical systems.