Analysis of safety systems with on-demand and dynamic failure modes

Leila Meshkat • University of Virginia • Charlottesville
Joanne Bechta Dugan • University of Virginia •Charlottesville
John D. Andrews • Loughborough University • Loughborough

Keywords: Dynamic fault trees, Safety Systems, On-demand failure, Markov models

SUMMARY & CONCLUSIONS

An approach for the reliability analysis of systems with on demand, and dynamic failure modes is presented. Safety systems such as sprinkler systems, or other protection systems are characterized by such failure behavior. They have support subsystems to start up the system on demand, and once they start running, they are prone to dynamic failure. Failure on demand requires an availability analysis of components (typically electromechanical components) which are required to start or support the safety system. Once the safety system is started, it is often reasonable to assume that these support components do not fail while running. Further, these support components may be tested and maintained periodically while not in active use. Dynamic failure refers to the failure while running (once started) of the active components of the safety system. These active components may be fault tolerant and utilize spares or other forms of redundancy, but are not maintainable while in use. In this paper we describe a simple yet powerful approach to combining the availability analysis of the static components with a reliability analysis of the dynamic components. This approach is explained using a hypothetical example sprinkler system, and applied to a water deluge system taken from the offshore industry. The approach is implemented in the fault tree analysis software package, Galileo

1. INTRODUCTION

Reliability analysis of safety systems, for example sprinkler systems or other protection systems, requires the consideration of two kinds of failure behaviors: failure on demand and failure during operation. That is, the system may fail to start when needed (on demand) or, once started, it may fail during use. The failure of the system to start when needed would indicate it's unavailability on demand; the failure once started indicates its unreliability during demand. Typically, the active components cannot be repaired/maintained during demand. The unavailability on demand of the system depends on the failure characteristics of its support components while in standby mode. These support components may be periodically tested and maintained while not in active mode. The unreliability during demand depends on the failure characteristics of the active components during demand. The reliability of the system is the probability that the system is available upon demand, and successfully achieves the mission operation during demand. In order to conduct a reliability analysis on such systems, we analyze each of the phases. The first phase is when the system is in standby mode, and the second phase, when the system is operational. This would require an availability analysis of the support subsystem in standby mode and a reliability analysis of the active components during demand.

2. MOTIVATING EXAMPLE

Let's consider a hypothetical example system. The system is composed of three sensors, two pumps, one digital controller, and each pump has a support stream composed of valves and filters. The sensors send signals to the digital controller and when temperature readings at two of the sensors are above threshold, the controller activates the pump. The system will be available upon demand, if at least two of the sensors are operational, and at least one of the pumps starts.

If a pump activates on demand, it means that the filters and valves in the pump stream are in working condition. At least one pump is needed for the system to operate. There is a backup pump which runs if the primary pump fails. System failure occurs if both the pumps are in a failed state.

Once a sprinkler system is activated, the sensors are no longer needed for reliable operation. However, at least one pump and digital controller must remain operational for a 10 hour period. Once the pump system is started, the pump stream is unlikely to fail during operation.

We can use a *dynamic fault trees*[6] to model the dependencies of the pumps on their pump streams. *Dynamic fault trees* extend traditional fault tree approaches by introducing special constraints to represent temporal relationships between various events. Functional dependencies are represented by *functional dependency constraints* [6]. The *functional dependency constraint* has a trigger input and one or more dependent inputs. When the trigger input occurs, the dependent inputs are forced to occur. A *cold spare gate* [3]can be used to model the backup pump which will get activated only if the primary pump fails.

Dynamic fault trees are translated into *Markov chains* for solution[5]. The fault tree model of our hypothetical example system is shown in figure 1. The cold spare gate(CSP) shows that pump 2 is a cold spare for pump 1, and is activated only if pump 1 fails. The functional dependency gates(FDEP) show the dependence of the pumps on their stream. The failure rate

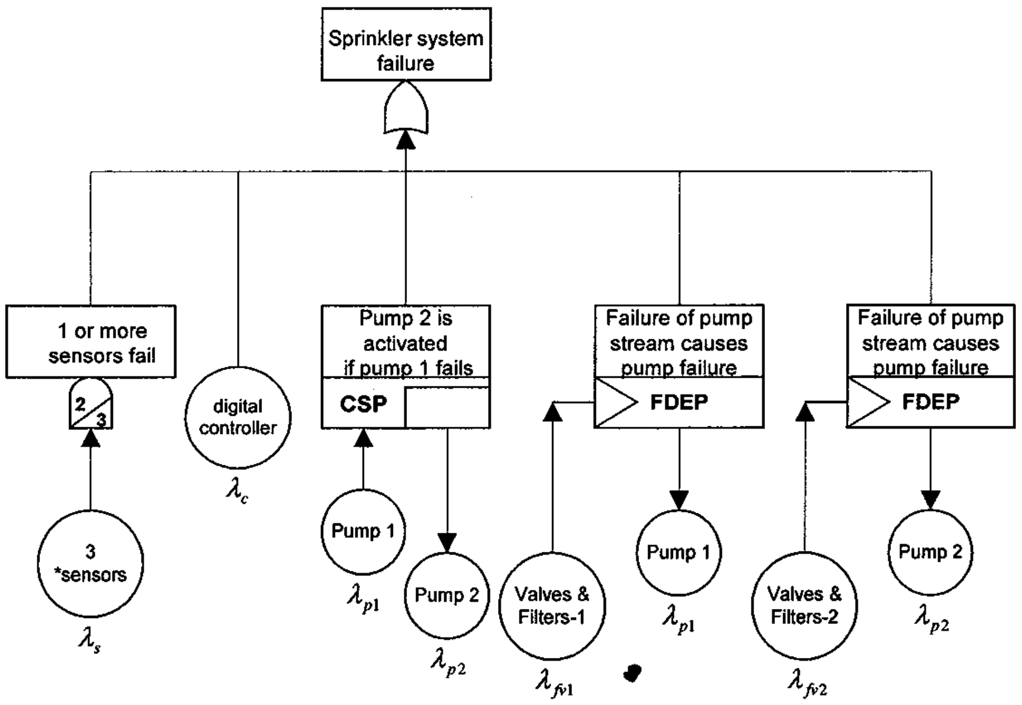
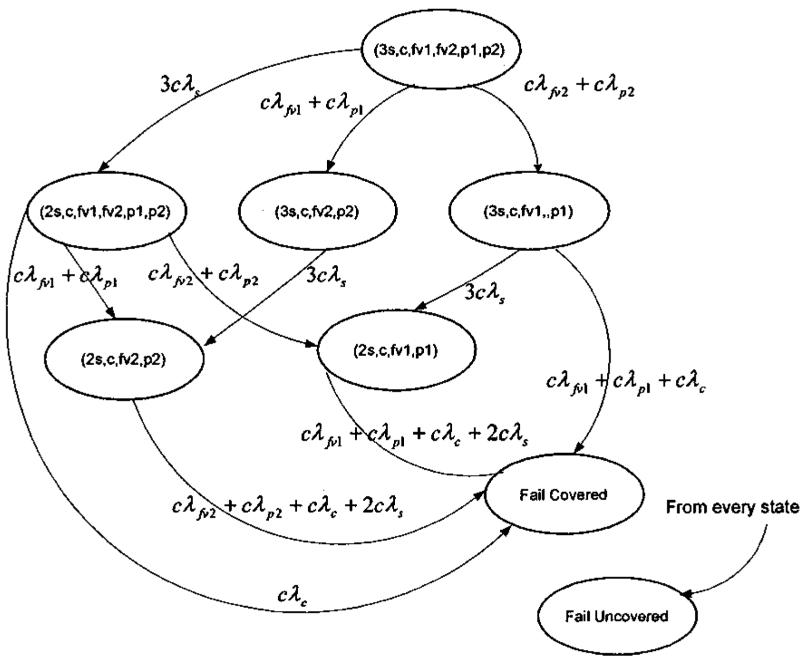


Figure 1: Fault tree for hypothetical example system using functional dependency gate



(3s,c,fv1,fv2,p1,p2)
$3c\lambda_s$
$c\lambda_{fv1} + c\lambda_{p1}$
$c\lambda_{fv2} + c\lambda_{p2}$
(2s,c,fv1,fv2,p1,p2)
(3s,c,fv2,p2)
(3s,c,fv1,,p1)
$c\lambda_{fv1} + c\lambda_{p1}$
$c\lambda_{fv2} + c\lambda_{p2}$
$3c\lambda_s$
$3c\lambda_s$
(2s,c,fv2,p2)
(2s,c,fv1,p1)
$c\lambda_{fv1} + c\lambda_{p1} + c\lambda_c$
$c\lambda_{fv1} + c\lambda_{p1} + c\lambda_c + 2c\lambda_s$
$c\lambda_{fv2} + c\lambda_{p2} + c\lambda_c + 2c\lambda_s$
Fail Covered
From every state
$c\lambda_c$
Fail Uncovered

Figure 2: Corresponding Markov chain for hypothetical example system.

of each component is shown underneath the basic events corresponding to the component in the figure 1. The Markov chain corresponding to the fault tree in figure 1 is shown in figure 2. The parameter c is used to indicate *covered failures* [3]. *Covered failures* of components may or may not lead to system failure depending on the remaining redundancy of the system. *Uncovered failures*, however, always lead to immediate system failure.

Although the pump streams and the sensors are only used to start up the system, the Markov chain in figure 2 takes them into consideration throughout the analysis, and they create additional states in the corresponding Markov chain. Real

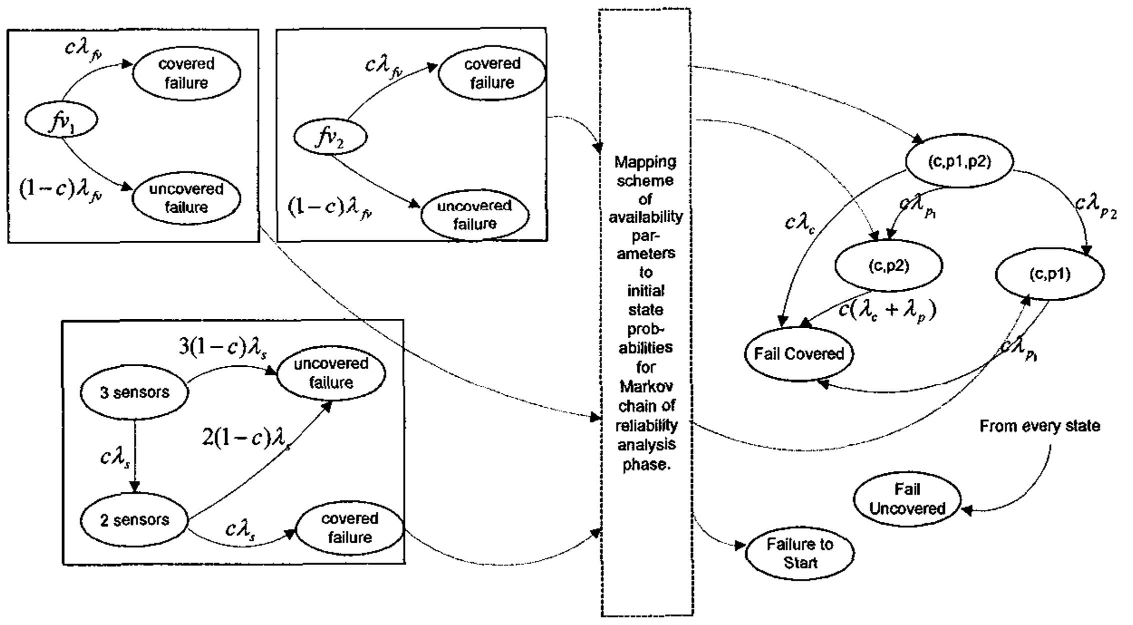


Figure 3: Separate models for pump streams, sensors, and dynamic subsystem

systems are more complex than our hypothetical example system, and state explosion is a problem often encountered when using Markov chains.

Moreover, in some instances, the components used to start up the system are also active during demand, but have different failure parameters. Different failure parameters cannot easily be modeled in this approach. Also, the support subsystem may be subject to repair and maintenance while the system is in standby mode. This is another attribute of the system that cannot be modeled using the above methodology.

A better approach would separate the availability analysis of the standby subsystem and the reliability analysis of the dynamic subsystem. For instance, we could create separate models for the analysis of each of the pump streams, as well as the sensors. This would be a viable approach since the pump streams and the sensors are independent. These separate models are shown in figure 3. This process entails the solution of three separate models, and the mapping of relevant information between these models. This could be a tedious procedure, if it is done manually, and without structure. It is for this reason that we define an approach and methodology for addressing this problem in the context of dynamic fault tree analysis. Furthermore, we create a module in the widely distributed reliability analysis software package, Galileo[7], which handles systems with dynamic and on-demand failure modes automatically. Our approach is explained in more detail in the following sections.

3. GENERAL APPROACH USING DYNAMIC FAULT TREE

3.1 Overview of approach

The approach that we present in this paper divides the reliability analysis of our hypothetical example system into two separate phases: availability analysis of the support system in standby mode, and reliability analysis of the system in dynamic mode. The steady state availability of the support components is obtained in the availability analysis phase. This steady state unavailability then determines the initial state probabilities of the Markov chain that only consists of the components that are in active mode during demand.

This simple yet powerful approach automatically combines the availability analysis of the system while in static mode with a reliability analysis of the system in active mode. The suggested approach is summarized in figure 3. Although this approach is fairly intuitive for a Reliability Engineer, it is not always easy to implement manually. This is because most systems are too complex for manual solution. We define and implement a new module in the widely distributed fault tree analysis software tool, Galileo[7]. This module models systems with on-demand and dynamic failure modes and solves for the reliability parameters, using this approach.

The work presented in this paper builds on the preliminary approach to a similar problem developed by Andrews & Ridley[2]. We have adapted the approach in [2] and engineered it to work within the modular dynamic fault tree analysis framework presented in [8] and[9], and implemented in Galileo[7].

3.2 Demand dependency gate

Within the context of a dynamic fault tree model, we define a new constraint to model failure on demand. The first input (or trigger input) to the *demand dependency constraint* is a fault tree describing the causes of failure on demand (or failure to start). This input can be either a static or dynamic fault tree. Components in this subtree are characterized by failure parameters, repair rates and/or maintenance intervals. T

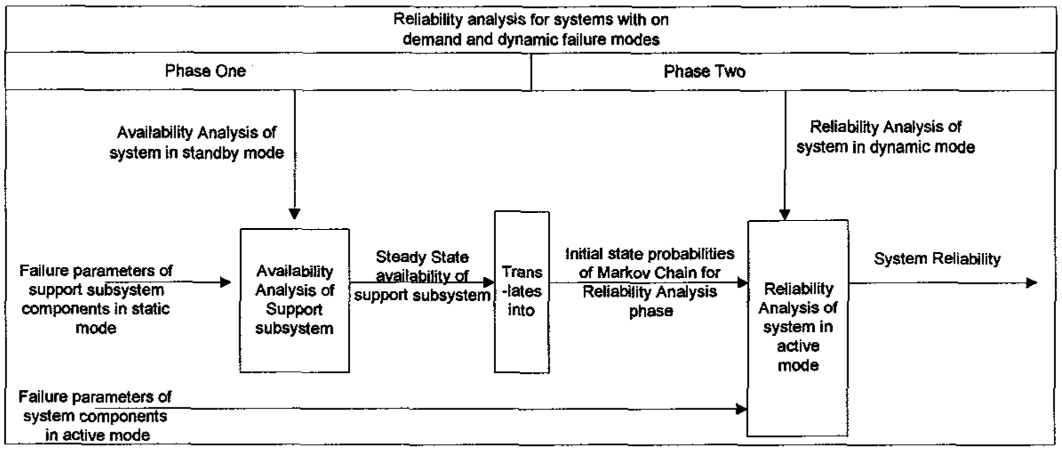

Reliability analysis for systems with on demand and dynamic failure modes
Phase One
Phase Two
Availability Analysis of system in standby mode
Reliability Analysis of system in dynamic mode
Failure parameters of support subsystem components in static mode
Availability Analysis of Support subsystem
Steady State availability of support subsystem
Trans -lates into
Initial state probabilities of Markov Chain for Reliability Analysis phase
Reliability Analysis of system in active mode
System Reliability
Failure parameters of system components in active mode

Figure 4: Overview of General Approach

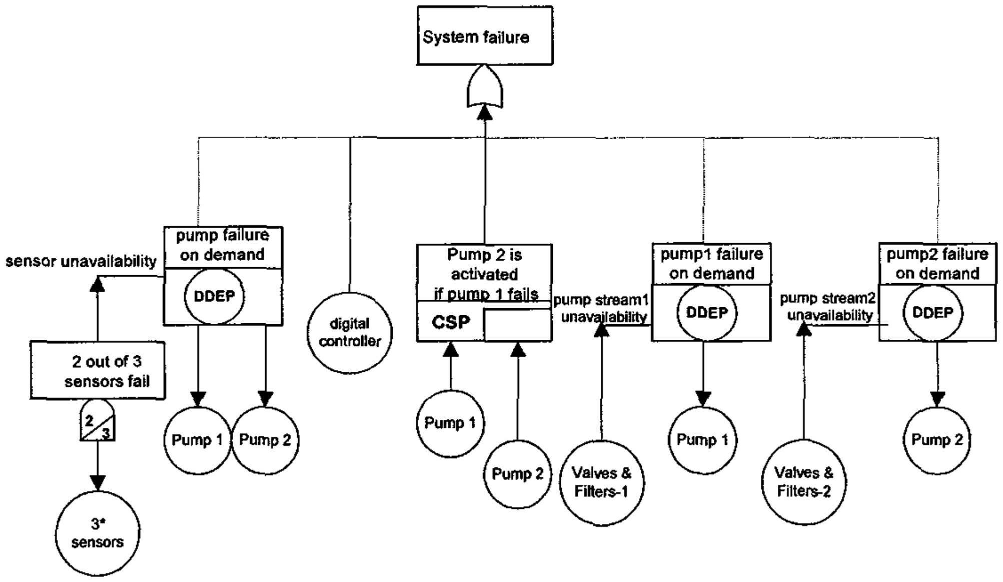
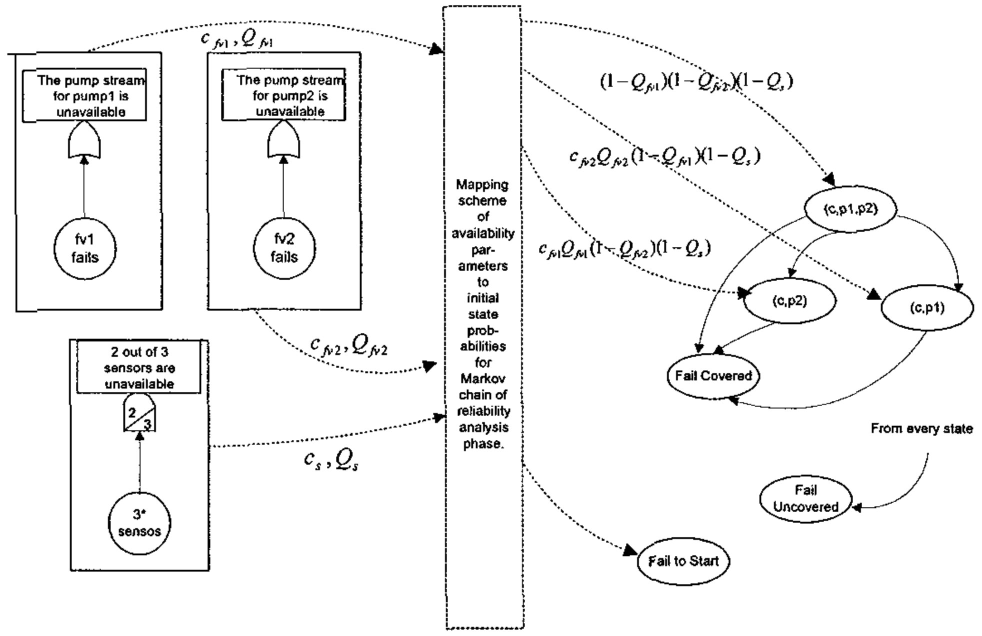


Figure 5: Fault tree for hypothetical example system using Demand Dependency Constraints

sub-tree is solved for the steady-state unavailability, which represents the probability that the subsystem is unable to start the safety system when demanded. The remaining inputs are dependent events which represent those components of the system whose functionality depends on the availability on demand of the subsystem represented by the trigger input. When the trigger input is unavailable, the dependent inputs are forced to occur. Therefore, the probability of occurrence of these inputs upon demand corresponds to the unavailability of the subtree representing the trigger input. For instance, consider a pump that depends on the availability of its pump stream in order to operate. The pump stream which consists of filters and valves would represent the first input to the *demand dependency constraint*, and the pump would be the dependent event.

3.3 Modeling hypothetical example system using the demand dependency constraint

The fault tree model of the hypothetical example system using the Demand Dependency Constraint can be seen in figure 5. The dependence of the system startup on the correct functionality of at least two of the sensors is shown by the first demand dependency constraint. If these sensors are unavailable at the time of demand, the pumps become unavailable, and hence the system cannot start up. On the other hand, the pumps also need their respective pump streams to be working in order to start up. This is shown by the other demand dependency constraints. Pump 2 is a backup pump, and is activated only if pump1 is in a failed state. Therefore, if the pump stream for pump1 is unavailable, but the pump stream for pump2 is available, and the sensors



The pump stream for pump1 is unavailable
fv1 fails
The pump stream for pump2 is unavailable
fv2 fails
2 out of 3 sensors are unavailable
3* sensos
c_{fv1}, Q_{fv1}
c_{fv2}, Q_{fv2}
c_s, Q_s
Mapping scheme of availability par-ameters to initial state prob-abilities for Markov chain of reliability analysis phase.
$(1-Q_{fv1})(1-Q_{fv2})(1-Q_s)$
$c_{fv2}Q_{fv2}(1-Q_{fv1})(1-Q_s)$
$c_{fv1}Q_{fv1}(1-Q_{fv2})(1-Q_s)$
{c,p1,p2}
(c,p2)
(c,p1)
Fail Covered
From every state
Fail Uncovered
Fail to Start

Figure 6: Mapping scheme for initial state probabilities of the reliability analysis phase

are available, the system will start up using pump2, instead of pump 1.

The separate subsystems of our hypothetical example system are modeled using Markov chains in figure 3. However, this is only for clarification purposes, and the steady state unavailability of the pumps can also be computed by using combinatorial methods, such as *Binary Decision Diagrams (BDD's)[4]*. This is because the subsystems corresponding to the pump streams and sensors are static fault trees. Modeling static subsystems using BDD's makes our solution procedure more efficient.

Our approach for solving this problem is to initially find the steady state unavailability of the sensors and the pump streams. If the sensors are unavailable, then the system cannot start, and the initial state for the Markov chain corresponding to the reliability analysis would be a failed state. If either of the pump streams cannot start, then their respective pumps will be unavailable, and the initial state of the Markov chain can be found accordingly. The initial state to the Markov chain is the state with all the components up only if the sensors and the pump streams are all available upon demand.

The approach can be seen in figure 6. The unavailability of pump streams and sensors are computed by conducting an availability analysis on their corresponding fault tree models. The unavailability measures are then used to find the initial state probabilities of the Markov chain corresponding to the reliability analysis phase.

4. SAFETY SYSTEM DESCRIPTION (WATER DELUGE SYSTEM)

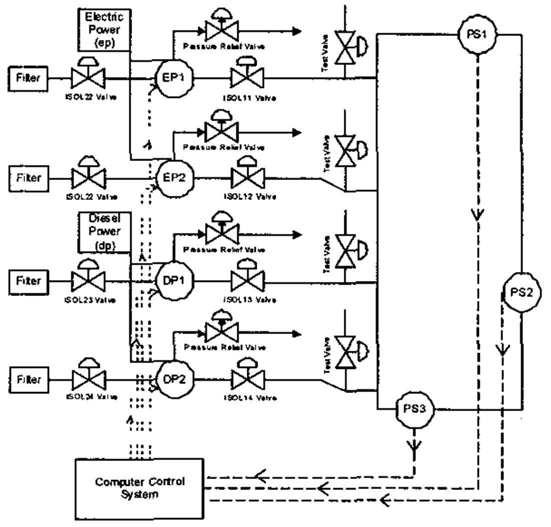


Figure 7: Schematic representation of the deluge system pump stream

The water deluge system[3] used to illustrate the dependency methodology is shown in figure 7. The features of this system are typical of water spray systems used in many different off-shore industries. Four pumps are used to provide the water demand to the ringmain. The ringmain transports the water round the platform to the take-off points where it is used to protect against the hazards posed by hydrocarbon fires and

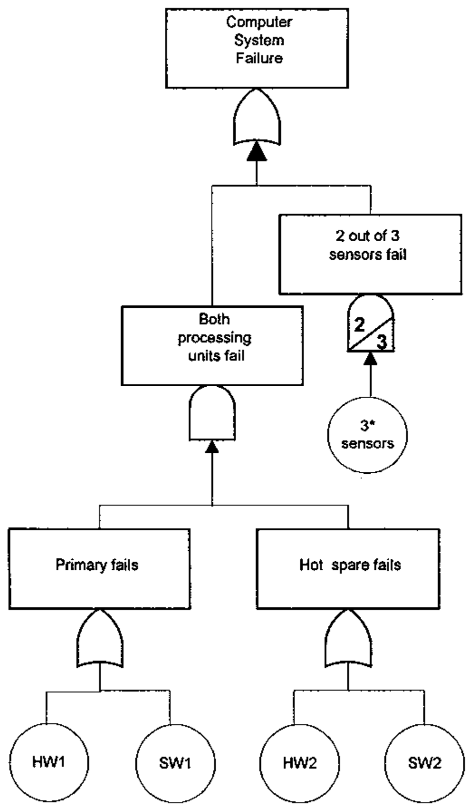


Figure 8: Fault tree model of computer system in WDS

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 (i.e. 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.

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 a dynamic component and can also fail once running. System failure will occur if fewer than two of the four streams can be activated (i.e. 3 from 4 fail).

5. FAULT TREE MODEL OF EXAMPLE SAFETY SYSTEM

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 that 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[1]. Modeling uncovered faults is crucial to the analysis of a fault tolerant computer system, and is discussed in more detail in[4] and [5]. A fault tree model showing the failure of the computer system is shown in figure 8, in which the basic events represent hardware (processors), software and the sensor set.

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, 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 that arise from the use of spares.

A spare gate is one of several dynamic gates introduced in [5] and it is used to model several dependencies associated with

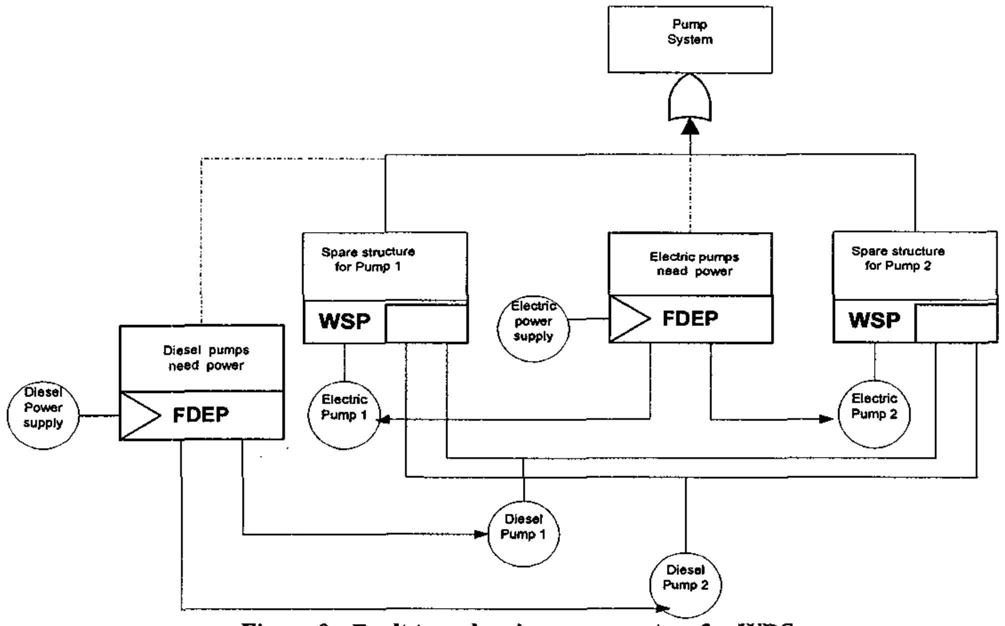


Figure 9: Fault tree showing pump system for WDS

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 to 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.

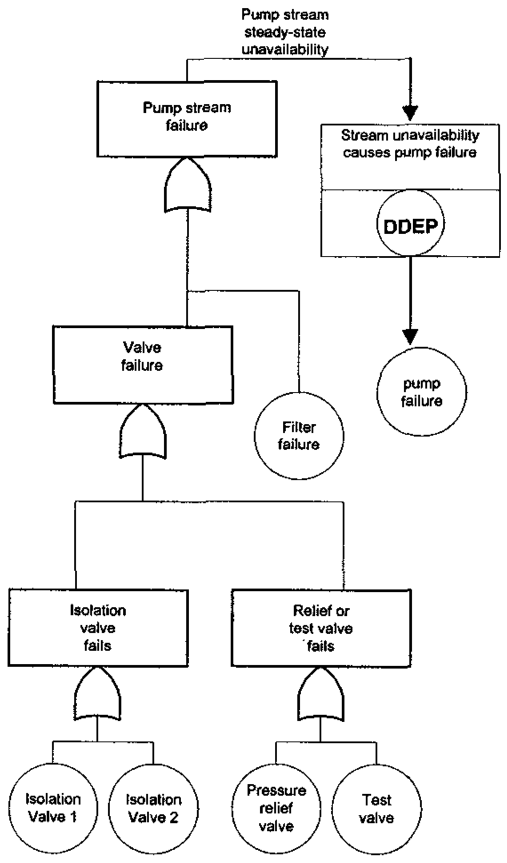


Figure 10: Fault tree showing pump stream for WDS

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 labeled EP1 and EP2 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[5]. 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. This is shown in the fault tree in figure 9.

Using the *demand dependency constraint,* we separate the static analysis of the pump stream from the dynamic analysis of the pumps themselves. The *demand dependency constraint* can be used to model the dependence on demand of the pumps on the pump stream. The pump stream is the trigger event and the pump is the dependent event. The implication of this constraint is that the pumps can fail to start on demand due to the unavailability of the pump stream. Since the unavailability of the pump stream at the moment that demand occurs determines whether or not the corresponding pump is available, the demand dependency constraint is designed to reflect a snapshot of the pump stream. The corresponding fault tree can be seen in figure 10.

The advantage of this approach is that it reduces the number of states in our Markov chain. The Markov chain which is used to solve this system doesn't need to take into account the valves and filters. Since the pump streams are unlikely to fail once the pump is running, it is not necessary to model each filter and valve in the reliability analysis phase. It is sufficient to know whether the stream is available on demand. The probability that the stream is available on demand is determined for each stream, and is used to determine the initial state probabilities for the Markov analysis of the pumps and power supplies in the reliability phase.

7. REFERENCES

[1] I. Lee and R.K. Dyer, "Faults, Symptoms and Software Fault Tolerance in the Tandem GUARDIAN90 Operating System," *Proceedings of the 23rd International Symposium on Fault Tolerant Computing,* June 1993

[2] J.D. Andrews and L.M. Ridley, "Analysis of systems with standby dependencies", *Proceedings of the 16th International System Safety Conference*, pg.: 80-88, Sept. 1998.

[3] J.D.Andrews, and J.B. Dugan, "Dependency Modeling Using Fault Tree Analysis", *17th International System Safety Conference,* August 1999.

[4] Joanne B. Dugan, and Stacy A. Doyle, "Incorporating imperfect coverage into binary *decision* diagrams", *Euro. J. Automat*, 30(8), 1996

[5] 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.

[6] 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.

[7] 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).

[8] 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

[9] Joanne Bechta Dugan, Bharath Venkataraman, Rohit Gulati, "DIFTree: A software package for the analysis of dynamic fault tree models" *Proceedings of the Reliability and Maintainability Symposium*, January 1997

8. BIOGRAPHIES

Leila Meshkat
Systems Engineering
University of Virginia
Charlottesville, VA
Email: leila@virginia.edu

Leila Meshkat is a Ph.D. candidate at the Department of Systems Engineering of the University of Virginia. Her research interests include Reliability and Risk analysis.

Joanne Bechta Dugan
Electrical Engineering
University of Virginia
Charlottesville, VA 22903
Email: jbd@virginia.edu

Dr. Dugan is a Professor of Electrical & Computer Engineering at the University of Virginia. She performs and directs research on the development and application of techniques for the analysis of computer systems that are designed to tolerate hardware and software faults.

John D. Andrews
Mathematical Sciences
Loughborough, LE11 3TU,
England
Email: J.D.Andrews@lboro.ac.uk

Dr. Andrews is a Senior Lecturer in the Department of Mathematical Sciences at Loughborough University. His current research interests concern the assessment of the safety and risks of potentially hazardous industrial systems.