This item was submitted to [Loughborough's Research Repository](#) by the author.
Items in Figshare are protected by copyright, with all rights reserved, unless otherwise indicated.

Maintenance modelling for computer-based systems

PLEASE CITE THE PUBLISHED VERSION

PUBLISHER

© IMechE / Professional Engineering Publishing

LICENCE

CC BY-NC-ND 4.0

REPOSITORY RECORD

Meshkat, Leila, Joanne Bechta Dugan, and J.D. Andrews. 2008. "Maintenance Modelling for Computer-based Systems". figshare. <https://hdl.handle.net/2134/3816>.

This item was submitted to Loughborough's Institutional Repository (<https://dspace.lboro.ac.uk/>) by the author and is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
<http://creativecommons.org/licenses/by-nc-nd/2.5/>

Maintenance modelling for computer-based systems

L Meshkat¹, J B Dugan^{2*} and J Andrews³

¹Department of Systems Engineering, University of Virginia, Charlottesville, Virginia, USA

²Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, Virginia, USA

³Department of Mathematical Sciences, Loughborough University, Leicestershire, UK

Abstract: A framework is presented for incorporating maintenance into a dependability analysis methodology for computer-based systems. Two types of maintenance are considered: failure-driven maintenance and time-driven maintenance. Failure-driven maintenance or repair is carried out when the system (or component) performance deviates from its expected performance and consists of all tasks performed to restore the functional capabilities of failed items, principally diagnosis and repair. Time-driven or scheduled maintenance is conducted on a specific time schedule in order to prevent system failure.

There may be dependencies between different components of a system with regard to their maintenance plans. These dependencies arise either because a component has maintenance priority over one or more components or because the maintenance of a certain component implies the maintenance of other components. Constructs are presented for modelling these dependencies in the context of dynamic fault tree analysis and a methodology is developed for solving the fault tree. The dynamic fault tree constructs effectively capture the failure dependencies between components. The approach is illustrated with an example based on a water deluge system.

Keywords: maintenance, dependability, computer-based systems, dynamic fault tree, dependency

NOTATION

f_i	failure rate of component i
r_j	repair rate of component j
D_k	probability of detecting the failure of component k
hw1	processor (hardware) component
hw2	processor (hardware) component in spare
v1	valve component 1
v2	valve component 2

1 INTRODUCTION

In broad terms, all operations that are conducted on a system in order to retain its components and keep it in

good condition are considered to be maintenance. The state of a system depends on the failure characteristics of its components and the maintenance plan which is conducted on the system. Computer-based systems are often upgraded periodically, as part of their maintenance plan. The upgraded versions often have different failure characteristics. Therefore it is necessary to take the maintenance plan into consideration in order to obtain an accurate dependability model of the system.

Maintenance for computer-based systems can be classified according to the reason why it is conducted. With regard to this classification scheme, maintenance can be *failure-driven* or *time-driven*. Failure-driven maintenance occurs when the system is maintained upon detection of a component failure. Time-driven or scheduled maintenance is performed on a predetermined schedule. A maintenance plan may combine both types of maintenance, repair some components upon their failure and maintain some components strictly on a fixed schedule.

An example water deluge system is explained in the next section. First there is a presentation of the dynamic fault tree constructs to model failure dependencies and later the constructs to model maintenance are added.

The MS was received on 11 July 2000 and was accepted after revision for publication on 3 August 2001.

**Corresponding author: Department of Electrical and Computer Engineering, School of Engineering and Applied Science, University of Virginia, Thornton Hall, Charlottesville, VA 22903-2442, USA.*

2 SAFETY SYSTEM DESCRIPTION (WATER DELUGE SYSTEM)

The water deluge system [1] used to illustrate the dependency methodology is shown in Fig. 1. 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 ring-main. The ring-main 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 ring-main 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 ring-main pressure will drop. Ring-main pressure is monitored and transmitted to the computer control system by the three pressure transmitters (PS1 to PS3). When two of the three transmitters indicate a low ring-main 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 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 back-ups.

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 on either side of the pump.

of the pump for maintenance purposes. 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 while running. System failure will occur if fewer than two of the four streams can be activated (i.e. three of four fail).

3 FAULT TREE MODEL OF EXAMPLE SAFETY SYSTEM

The computer control system consists of the three pressure sensors (of which two are needed), plus the hardware and the software. The hardware consists of redundant processors in hot standby mode, each equipped

Fig. 1 Schematic representation of the deluge system pump stream

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 the case of error. When an error is detected, control is switched to the back-up 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 back-up. This latter case (undetected error) is an example of an uncovered (or unrevealed) fault, which leads to immediate system failure. Another example of an uncovered fault is a software fault that affects both processors simultaneously. It might be expected, 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 [2]. Modelling uncovered faults is crucial to the analysis of a fault-tolerant computer system, and is discussed in more detail in references [3] and [4]. A fault tree model showing the failure of the computer system is shown in Fig. 2, 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, the pump streams and power supplies will be ignored; attention will be concentrated 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 modelling techniques. A pump which is in use experiences a different failure rate than one in standby. Therefore, there is a need to keep track of which pumps are being used and which are in standby. A spare gate is used to model the failure dependencies that arise from the use of spares. A spare gate is one of several dynamic gates introduced in reference [5] and it is used to model several dependencies associated with the use of spares.

Firstly, a component which is used as a spare has an associated *dormancy factor*, α , 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 modelled 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 being 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. When modelling pooled spares there is a need to keep track of not only the state of each component but also the order in which they have failed, in order to determine which spare is being used where. Further, it might be the case that components have preferences for replacements, in that there is a 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 Fig. 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 labelled EP1 and EP2 represent the two electric pumps, which are both initially active (on

Fig. 2 Fault tree model of the computer system in the water deluge system

Fig. 3 Fault tree showing the pump system for the water deluge system

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 there is a preference for EP2 to be replaced by DP2 then the order of DP1 and DP2 inputs on the second spare gate could be switched.

Next the power supplies will be considered. 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 modelled 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 Fig. 3.

Using the *demand dependency construct* [6], the static analysis of the pump stream can be separated from the dynamic analysis of the pumps themselves. The demand dependency construct 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 construct 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 the corresponding pump is available, the demand dependency construct is designed to reflect a snapshot of the pump stream. The corresponding fault tree can be seen in Fig. 4.

4 PROBLEMS THAT NEED TO BE ADDRESSED

In order to find the dependability measures of the system, first the availability measures of the system components at the time that a demand occurs need to be found and then the probability that the system performs satisfactorily for the duration of demand. The problem that is addressed in this paper is of finding the availability measures of the system components at the time that demand occurs. These availability measures are used to find the dependability measures of the entire system, which is done automatically using the DDEP gate as defined in reference [7].

Fig. 4 Fault tree showing the pump stream using the DDEP gate for the water deluge system

The issues involved in finding the availability measures of the system components at the time of demand are the following:

1. The system components are periodically maintained. Once scheduled maintenance is conducted on the system, all the components are brought up to an 'as good as new' situation.
2. In addition to scheduled maintenance at fixed points in time, the system components are also surveyed continuously and repaired if a failure is detected in

them. However, the probability of detecting a failure in a survey is not one.

3. There is dependency in the maintenance of some components. If a failure is detected in any of the valves, all of them must be maintained. This is an example of a *maintenance implication*. A maintenance implication arises when the maintenance of one component implies that other components in the same group will also be maintained. Further, the processor has priority over its spare; i.e. the processor must be maintained first if both the processor and its spare are

in a failed condition. This is an example of a *maintenance priority*. A maintenance priority is used to specify that the maintenance of one component must precede the maintenance of another, should they both need repair.

These are issues that need to be addressed in the case of the water deluge system. In the following section, a description is given of the issues that exist in the general case for the maintenance of computer-based systems, and the approach used to address these issues.

5 GENERAL APPROACH TO THE MAINTENANCE MODELLING OF COMPUTER-BASED SYSTEMS

5.1 Maintenance schemes

In this section, the several possible maintenance schemes are discussed. These maintenance schemes can relate to the maintenance behaviour of one component or they can imply some kind of dependence between two or more components. In the case where the behaviour implies some sort of dependence between one or more components, constructs can be defined in the context of dynamic fault trees to connect the components in question together and show the behaviour. These constructs are then incorporated into the fault tree of the system. Also, the independent maintenance behaviour of components are considered in the component characteristics of a fault tree. Thus the fault tree can represent the maintenance behaviour of the system in addition to its failure behaviour.

5.1.1 Priority issues

It may be necessary to conduct maintenance on one or more components before a component in question can be maintained; i.e. one component may be more critical than another and thus has higher priority. This situation is modelled using the construct in Fig. 5. In this figure, two different situations have been shown. The left-hand side of the figure shows that component C cannot be maintained before component B is maintained, and component B cannot be maintained before component A. The right-hand side shows that component A needs to

Fig. 5 Maintenance priority constructs

be maintained before all other components of the system. Priority issues create dependencies between components.

5.1.2 Maintenance implications

Sometimes the system structure might be such that it would be meaningless to repair a certain component without maintaining some other component. This indicates a repair dependency between the two components in question. Therefore, the maintenance or repair of a component would imply the repair or maintenance of some other component. This relationship could be mutual between two components, i.e. the repair of any of the two components could imply the repair of the other component. Maintenance implication constructs are shown in Fig. 6. The *repair implication* construct between components A and B shows that component B must be repaired every time component A is repaired and the *repair condition* construct shows that the repair of either of the components A or B implies the repair of the other.

5.1.3 Upgrading components

Computer-based systems can be upgraded periodically. The system failure parameters can change after the upgrade since the new upgraded version is likely to have different failure parameters. Therefore it is important to keep track of the upgrade information. If an upgrade changes the failure parameter of the up-graded component, but does not impact other components or create any kind of dependencies between components, it can be considered a component characteristic, and it is not necessary to show it with a fault tree construct. If an upgrade creates dependencies, e.g. if two software components must be upgraded simultaneously, then the implication construct can be used to capture this behaviour.

5.1.4 Maintenance frequency

Different components of a system may have various maintenance schedules. Some components may be

Fig. 6 Maintenance implication constructs

maintained only upon failure, while other components may be maintained on a predetermined schedule. The maintenance frequency of each component is a characteristic of that particular component and does not impact other components. Therefore this behaviour is captured in the component characteristics.

5.1.5 Summary

The maintenance plan for a system can be expressed as a combination of the schemes mentioned above. The present approach is to divide the entire maintenance plan into its various schemes and then consider each scheme separately and include it in the system model. The solution of this model yields the dependability measures of the maintainable system.

5.2 Analysis of maintainable systems

The problem addressed in this section is the solution of the fault tree structures developed in the previous section. The approach used here is to modularize the maintainable system fault tree and solve each of the independent modules for their dependability measures and then combine the measures of each module to obtain the system dependability. The approach therefore consists of three consecutive steps:

1. Modularize the maintainable system fault tree into its independent subtrees.
2. Solve each independent module for its reliability (availability) measures.
3. Combine the results to obtain the system dependability measures.

5.2.1 Modularization

The modularization of the maintainable system fault tree entails finding subtrees that are independent in their failure characteristics as well as their maintenance characteristics. Two subtrees are considered independent in their failure characteristics if they do not have any common basic events. The most efficient and simple algorithm is the approach presented by Dutuit and Rauzy [8]. This is a linear-time algorithm that can detect modules in fault trees.

Two modules are defined to be independent in their maintenance characteristics if they are not connected with either of the 'maintenance priority' or 'maintenance implication' constructs. If they are connected by any of these gates then their maintenance characteristics show dependencies, and they must therefore be modelled together.

5.3 Solution of independent modules

The state of a system at any point in time depends on the maintenance scheme that is administered on that system. Often the maintenance plan is designed to optimize the reliability of the system [9]. System availability and total cost are also some of the criteria that are considered during the maintenance design stage [10, 11].

The failure behaviour of the system components can be predicted based on the component characteristics and the maintenance plan. The dependability of the system can then be obtained based on the failure behaviour of the system components. This is done by obtaining the repair rate and characteristics of the individual components using the maintenance plan, and incorporating them into the reliability model of the system.

Multiple phases of operation are observed in a system that undergoes scheduled maintenance [12]. This is because the periodic scheduled maintenance can cause phase behaviour in a system, since at times of maintenance the system state changes abruptly as component failures are detected and repaired or upgraded. Therefore, the modelling techniques used for modelling multiple-phased systems may also be applied to systems that undergo maintenance [6]. The dependability measures of a system undergoing maintenance can be obtained once the failure and maintenance characteristics of the system components have been identified. The following dependability measures for maintainable systems are considered here:

1. The steady state or limiting pointwise availability of the system

$$a = \lim_{t \rightarrow \infty} a(t) \quad (1)$$

2. Average availability over a maintenance cycle T . Thus $\bar{a}_T$ is defined for a maintenance cycle (time period between maintenance actions) T as

$$\bar{a}_T = \frac{1}{T} \int_0^{T-M} a(t) dt \quad (2)$$

where M is the amount of time needed to perform the maintenance on the system.

In this section, several of the schemes that occur in systems undergoing maintenance are considered and these schemes are classified according to the solution technique that would apply to them.

5.3.1 Scheduled maintenance and upgrade

Scheduled maintenance and system upgrade can be modelled using the techniques applied to the modelling of multi-phased systems with fixed durations [5, 13–15]. This is done by considering the scheduled maintenance as

instantaneous repairs at the time of maintenance checks. In the context of Markov modelling, the scheduled maintenance can be modelled by readjusting the state occupation probabilities after maintenance. In a broad sense, the probabilities associated with the state in which the maintained component is down should be allocated to the state where everything else is identical to the former state but the maintained component is up. If there are failure sequence dependencies or minimal repair, then it is necessary to take that into consideration in readjusting the state occupation probabilities.

In the case of a system upgrade, not only will the component in question be brought up to a perfect state but its failure parameters may also change. This is because the newer version of the component may have a different failure rate. This is another attribute that should be taken into consideration in the system Markov chain.

5.3.2 Repair upon failure

Where components (modules) are repaired upon failure, if the repair of the component (module) in question is independent, with an exponential distribution, then it is possible to obtain the availability of that component (module) by solving the Markov model in Fig. 7. In this figure, the Markov model has two states, one is the operational state (Up) and the other is the failed state (Down). The failure rate of the component is f and r is the repair rate. Independent repair of a component (or module) implies that there are no maintenance dependencies between that component (or module) and the rest of the system components. A module may consist of one or more components. In Fig. 7, the module has only one component.

If the repair of several components are interdependent, then they cannot be analysed in separate models. They should be modelled in a combined Markov model, and their respective availability could then be obtained from the solution of that Markov chain. Figure 8 shows the Markov chain related to the hardware components (remember that the first hardware had maintenance priority over the second one). The state (hw1, hw2) is where both of the hardware components are functional. Hardware 1 (hw1) fails with a rate of f_{hw1} . Upon the failure of hardware 1, the system enters the state (hw2) in which only hardware 2 is functional. There is a probability of D_{hw1} that the failure of hardware 1 will be detected. If it is detected, it will be repaired with a rate

Fig. 7 Markov model for independent repair of components

Fig. 8 Markov chain for the two hardware

of r_{hw1} . As can be seen in the figure, when both of the hardware are in a failed state, hardware 1 is the first to be maintained.

Figure 9 is the Markov chain for the valves. Scheduled maintenance is always performed on both valves together. However, if one fails and the failure is detected, it is repaired and if both of them are failed, and their failures are detected, they are repaired together. Note that this is different from the hardware systems. In the case of the hardware systems, the first hardware is repaired first when both of the hardware systems are in a failed state.

5.3.3 Dealing with priority and implication issues

The priority and implication issues must be considered while readjusting the state occupation probabilities after maintenance. For instance, if component A must be maintained before component B can be maintained, then the state probabilities need to be readjusted so that the probabilities of all states that have A or B components down are allocated to the states that are identical in other respects but the A and B components are both up.

6 SYSTEM-LEVEL SOLUTION

Once the availability measure of each independent module within a fault tree has been obtained, it is easy to find the availability measure for the entire system. This

Fig. 9 Markov chain for the valves

Fig. 10 Expected availability of a scheduled maintenance system with a maintenance period of T

is done by considering each of the independent modules as a basic event in the system fault tree with an availability measure equal to that obtained from the availability analysis of the module. The fault tree is then solved as usual for the availability measure of the entire system. It is important to note that the availability measures used for all the modules must be the same measure. It should be either the steady state availability of each module or the average availability of each module.

7 SOLUTION OF THE WATER DELUGE SYSTEM

In this section, an availability analysis is conducted of the water deluge system presented in Section 2. This analysis is based on determining a maintenance schedule for the system while it is in standby mode and finding the steady availability measures of the system.

Safety systems and other protection systems are often in standby mode for long periods of time. They are maintainable during this time period and demand may occur at any random point of time. The availability of the system is at its maximum immediately after its scheduled maintenance. In fact, the fluctuations of the expected availability of the system follow a stochastically similar pattern in each period, which is identified by the time period between two maintenance intervals. Because it is assumed that demand may occur at any time during the interval, the average availability of the support system is the average availability of the system during a single period. Figure 10 shows the conceptual fluctuations of the average availability of a scheduled maintenance system over time.

The maintenance dependencies of the valves are shown in the fault tree of the pump streams 4 with repair condition constructs. Repair condition constructs imply

bidirectional maintenance implication constructs between their corresponding components. The corresponding Markov chain for the valves is shown in Fig. 9. The maintenance dependencies of the hardware systems are shown in the fault tree of the computer system with the maintenance priority construct 2. The Markov chain associated with the failure behaviour of the hardware systems is shown in Fig. 8. Since the first hardware has maintenance priority over its spare, once the system reaches a state in which both are failed, the first hardware is repaired first. Note that if the failures are not detected, the components will remain in a failed state until scheduled maintenance.

The failure behaviour of the independent components during each period, which is the time interval between two maintenance procedures, is shown in Fig. 11. It is assumed that if the failure of a component is detected, it is repaired at the time of detection. The failure rate is considered to be f and the repair rate is r . The probability of detecting the occurred failure is D . Note that the failed state of the components has been partitioned into failed detected and failed undetected states. If the component reaches the failed detected state, it will be repaired. If it

Fig. 11 Markov chain showing the behaviour of the components during each period

Table 1 Failure and repair rates for the water deluge system

Component name	Standby failure rate	Active failure rate	Detection probability	MTTR*
Processor	$10^{-5}/\text{h}$	$5 \times 10^{-4}/\text{h}$	0.5	3 h
Control software	0	$1.2 \times 10^{-5}/\text{h}$	0	1 day
Valves	$1.42 \times 10^{-5}/\text{h}$	$5.3 \times 10^{-4}/\text{h}$	0.2	4 h
Filters	$3 \times 10^{-6}/\text{h}$	$3 \times 10^{-6}/\text{h}$	0.1	6 h
Sensors	$1.27 \times 10^{-5}/\text{h}$	$3.4 \times 10^{-4}/\text{h}$	0.75	3 h
Diesel power supply	$2.3 \times 10^{-5}/\text{h}$	$2.3 \times 10^{-4}/\text{h}$	0	8 h
Electric power supply	$10^{-5}/\text{h}$	$10^{-4}/\text{h}$	0	6 h
Electric pump	$7 \times 10^{-6}/\text{h}$	$7 \times 10^{-4}/\text{h}$	0.5	1 day
Diesel pump	$7 \times 10^{-6}/\text{h}$	$7 \times 10^{-4}/\text{h}$	0.5	1 day

* MTTR, mean time to repair.

fails and the failure is undetected, it will not be repaired until scheduled maintenance occurs.

8 NUMERICAL SOLUTION OF THE WATER DELUGE SYSTEM

In this section a set of failure and repair rates is considered for each of the components of the water deluge system and the models developed in the former section are used to obtain the average availability of the water deluge system during a period of two weeks or 336 hours. Table 1 shows the active and standby failure and repair rates as well as the detection probability of the components of the water deluge system. If the failure of a component is detected, then it is repaired. Otherwise, the component remains in a failed state until scheduled maintenance occurs at the end of the two weeks.

The average availability of each component is obtained when in the standby mode. The time period T is equal to two weeks or 336 hours. The availability measures obtained for each component are given in Table 2.

9 CONCLUSIONS

This paper presents a framework for incorporation of maintenance into dynamic fault tree analysis. Different

maintenance schemes have been considered and constructs for modelling these schemes have been developed in the context of dynamic fault tree analysis. The solution of the maintainable system fault tree is obtained by finding the independent modules of the fault tree, solving each independent module for its availability measure and then combining the availability measures of the separate modules to obtain the system availability.

REFERENCES

- 1 Andrews, J. D. and Dugan, J. B. Dependency modeling using fault tree analysis. In Proceedings of the 17th International System Safety Conference, August 1999.
- 2 Lee, I. and Dyer, R. K. Faults, symptoms, and software fault tolerance in the tandem guardian90 operating system. In Proceedings of the 23rd International Symposium on Fault Tolerant Computing, June 1993.
- 3 Dugan, J. B. and Trivedi, K. S. Coverage modeling for dependability analysis of fault-tolerant systems. *IEEE Trans. on Computers*, 1989, **38**(6), 775–787.
- 4 Pecht, M. (Ed.) *Product Reliability, Maintainability and Supportability Handbook*, 1995 (CRC Press, Boca Raton, Florida).
- 5 Dugan, J. B., Bavuso, S. and Boyd, M. Dynamic fault tree models for fault tolerant computer systems. *IEEE Trans. on Reliability*, September 1992, **41**(3), 363–377.
- 6 Meshkat, L., Dugan, J. B. and Andrews, J. D. Analysis of safety systems with on-demand and dynamic failure modes. In Proceedings of the Annual Reliability and Maintainability Symposium, January 2000, pp. 14–21.
- 7 Meshkat, L. Dependency modeling and phase analysis for computer based systems. PhD thesis, University of Virginia, August 2000.
- 8 Dutuit, Y. and Rauzy, A. A linear-time algorithm to find modules in fault trees. *IEEE Trans. on Reliability*, September 1996, **45**(3), 422–425.
- 9 Zwingelstein, G. C. Reliability centered maintenance. In Proceedings of the 2000 Annual Reliability and Maintainability Symposium: Tutorial Notes, January 2000.
- 10 Lim, J.-H. and Park, D. H. Evaluation of average maintenance cost for imperfect repair model. *IEEE Trans. on Reliability*, June 1999, **48**(2), 199–204.
- 11 Pukite, J. and Pukite, P. *Modeling for Reliability Analysis*, 1998 (IEEE Press).

Table 2 Obtained availability measures for water deluge system components

Component	Availability
Processor 1	0.999 145 361
Processor 2	0.998 978 51
Valves	0.998 08
Filters	0.999 54
Sensors	0.999 44
Diesel pump	0.996 15
Electric pump	0.998 32
Diesel power supply	0.999 32
Electric power supply	0.999 32

- 12 **Sharma, T. C., Somani, A. K. and Palnitkar, S.** Reliability modeling of scheduled maintenance systems with latent failure. *Int. J. Reliability, Qual. Saf. Engng*, 1998, **5**(4), 359–371.
- 13 **Ritcey, J. A., Somani, A. K. and Au, S. H. L.** Computationally-efficient phased-mission reliability analysis for systems with variable configurations. *IEEE Trans. on Reliability*, December 1992, **41**(4), 504–511.
- 14 **Esary, J. D. and Ziehms, H.** Reliability analysis of phased missions. In *Reliability and Fault Tree Analysis: Theoretical and Applied Aspects of System Reliability and Safety Assessment* (Eds R. E. Barlow, J. B. Fussell and N. D. Singpurwalla), 1975, pp. 213–236 (SIAM, Philadelphia, Pennsylvania).
- 15 **Alam, M. and Al-Saggaf, U. M.** Quantitative reliability evaluation of repairable phased-mission systems using Markov approach. *IEEE Trans. on Reliability*, December 1986, **R-35**(5), 498–503.