Phased Mission Analysis Using the Cause-Consequence Diagram Method

by

Gintare Vyzaite

Master's Thesis

Submitted in partial fulfilment

of the requirements for the award of

Master of Philosophy

of Loughborough University

September 2007

ABSTRACT

Most reliability analysis techniques and tools assume that a system used for a mission consists of a single phase. However, multiple phases are natural in many missions. A system that can be modelled as a mission consisting of a sequence of phases is called a phased mission system. In this case, for successful completion of each phase the system may have to meet different requirements. System failure during any phase will result in mission failure. Fault tree analysis, binary decision diagrams and Markov techniques have been used to model phased missions.

The cause-consequence diagram method is an alternative technique capable of modelling all system outcomes (success and failure) in one logic diagram. The structure of the diagram has been shown to have advantageous features in both its representation of the system failure logic and its subsequent quantification, which can be applied to phased mission analysis.

The work developed outlines the use of the cause-consequence diagram method for systems undergoing non-repairable phased missions. Methods for the construction of the cause-consequence diagram for such systems are considered. The disjoint nature of the resulting diagram structure can be utilised in the later quantification process. The similarity with the Binary Decision Diagram method enables the use of efficient and accurate solution routines. The method is illustrated with the application of an example of the cause-consequence diagram method to a non-repairable phased mission system. The system considered is an aircraft flight. The technique is computationally efficient and the work presented here shows that it is superior to the binary decision diagram. The work is extended to systems that can have multiple faults (i.e, minor, which would allow the system to progress to the next phase, and major, which would cause system failure).

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Sarah Dunnett and my director of research Prof. John Andrews for their guidance over these years. Their input was invaluable.

I would also like to thank the colleagues and academics at the department of Aeronautical and Automotive Engineering for their help, advice and support in my project.

My deepest thanks go to my family, without whose support and encouragement during my studies I wouldn't be here today. Thank you for believing in me.

Finally, the greatest thank you I would like to say to Carlo, who was always next to me.

CONTENTS

NOMENCLATURE

C_i	Existence of a minimal cut set i
c_{k_j}	Event that component c_k works the phase k, given that it was functioning through all previous phases
$E_i(t)$	Expected number of times system enters state i
$ENF(t_0, t_i)$	Expected number of failure in time interval t_0 to t_1
$F(t)$	Cumulative failure distribution
$f(t)$	Failure probability density function
$G_i(\mathbf{q}(t))$	Criticality function for component i
I_i^{BP}	Barlow-Proschan measure of initiator importance
I_e^{BP}	Barlow-Proschan measure of enabler importance
I_i^{CM}	Criticality measure of importance
I_i^{FV}	Fussell-Vesely measure of component importance
$I_{C_i}^{FV}$	Fussell-Vesely measure of minimal cut set importance
I_i^{ST}	Structural measure of importance
$P(T)$	Event probability
p_{ij}	Component i availability in phase j
Q	Phased mission system unreliability
$Q_{sys}(t)$	System unavailability function
$q(t)$	Minimal cut set unavailability
q_{c_i}	Component unavailability
q_{w_i}	Weight of basic event i
R	System reliability i
U	Performance state indicator
$W(0,t)$	Expected number of system failures in time t
$W_T(t)$	Rate of failure

$w_s(t)$	System failure intensity
x_i	Binary indicator variable for component states
$\lambda(t)$	Component failure rate
$\mu_i(t)$	Component detection/repair rate
$\phi(x)$	System structure function
$\rho(x)$	Binary indicator function for each minimal cut set
θ	Inspection interval
θ_i	Occurrence of minimal cut set i
τ	Mean time to repair

1. INTRODUCTION

Many systems perform a mission, which can be divided into consecutive time periods - phases. In each phase, the system needs to perform a specific task. The system configuration, the phase duration, and the failure rates of components often vary from phase to phase. Burdick *et al* [1] describe a phased mission as a task to be performed by a system during execution of which the system is altered such that the logic model changes at specified times. Thus, during a phased mission, time periods (phases) occur in which either the system configuration, system failure characteristics, or both are distinct from those of any immediately succeeding phase.

The most important aim of phased mission analysis is to calculate the exact, or obtain bounds for, mission unreliability. This is defined as the probability that the system fails to function successfully in at least one phase. Estimating the mission unreliability by the product of the phase unreliabilities results in inaccuracies, since basic events are shared among logic models of the various phases which are not then independent. Esary and Ziehms [2] used a fault tree method for the analysis of the phased missions for non-repairable systems. They introduced basic event transformation and cut set cancellation techniques. But the method proposed by Esary and Ziehms was unable to calculate the probability of failure of each phase due to cut set cancellation, only of the whole mission. La Band and Andrews [3] introduced a new method based on non-coherent fault trees that determines the probability of failure of each phase in addition to the whole mission unreliability. The method combines the causes of success of previous phases with the causes of failure for the phase being considered to allow both qualitative and quantitative analysis of both phase failure and mission failure.

Zang, Sun and Trivedi [4] proposed an algorithm for analysis of phased mission systems based on binary decision diagrams (BDDs). Such diagrams give a representation of the system failure logic which is in a format more effective for analysis than that of a fault tree. As such, BDDs offer efficient mathematical manipulation, but are difficult to construct directly from the system definition and hence are generally obtained by converting from a fault tree. The method proposed by [4] only determines the unreliability of the whole mission. A BDD methodology was also applied by La Band and Andrews [3] to evaluate the probability of failure of each phase in the mission.

As both of these methods have their own drawbacks, another method for system reliability/unreliability was introduced Nielsen [5]. The cause-consequence diagram method was developed at RISO Laboratories, Denmark, as a graphical tool for analysing relevant accidents in a complex nuclear power plant. The method presents logical connections between causes of an undesired (critical) event and the consequences of such an event, if one or more mitigating provisions fail. As all consequence sequences are investigated, the method can assist in identifying system outcomes, which may not have been investigated at the design stage. Ridley and Andrews [6] notice that, for some types of system, the final cause-consequence diagram has an identical structure to that of the BDD. They noted, however that the cause-consequence diagram was more concise due to the automatic extraction of common independent sub-modules. As the cause-consequence diagram can be obtained directly from the system description, there was no need to develop and convert from a fault tree to BDD. They also noted that as the BDD is a more efficient tool than the fault tree method then the cause-consequence diagram formulation can be advantageous. The cause-consequence diagram also has the capability to model the failure of each phase in addition to the whole mission in one diagram.

In this work cause-consequence analysis is applied to phased missions. The fault tree analysis is reviewed in Chapter 2 and some approximation techniques used to evaluate system reliability/unreliability and importance measures are described. Chapter 3 reviews binary decision diagrams. The binary decision diagram method can suffer if the order in which components are considered is not well chosen and this results in an increase in the size of the resulting diagram. This decreases the efficiency of the method and hence many schemes have been devised to obtain the most efficient order of components. Some of the possible ordering schemes are discussed in Chapter 5. In Chapter 4 the cause-consequence diagram method is described and reviewed. Phased mission systems are described in Chapter 6. This chapter outlines the different methods available for evaluating a phased mission system. The purpose of this work was to apply the cause-consequence diagram method to the phased mission systems and this is presented in Chapter 7. Two methods for constructing a cause-consequence diagram for phased mission system are presented. The following chapter gives the construction and analysis of the cause consequence diagram of a specific phased mission system - an aircraft flight. It is a very complex system for the capacity of this theses therefore a simplified version of the system was used. Chapter 9 goes on to describe a modularisation technique that can be applied to the cause-consequence diagrams. The conclusions of the work and some suggestions for the future work are given in Chapter 10.

2. FAULT TREE ANALYSIS

2.1 Introduction

There are two main types of modelling tools used for reliability analysis. They are inductive, or forward, analysis, and deductive, or backward, analysis. An inductive analysis starts at component level and proceeds forward identifying the possible consequences. Fault tree analysis is an example of deductive analysis, where the process starts at a possible consequence and goes backwards trying to identify all possible causes. It provides a diagrammatic description of system failure in terms of the failure of its components.

2.2 Construction Of a Fault Tree

The first step in the construction of a fault tree is to determine a system failure mode. The system failure mode is termed the 'top event' and the fault tree is developed in branches below this event showing its causes. It is important that the definition of the top event is not too broad or too narrow to produce the results required. If the system has more than one failure mode, multiple fault trees would be constructed to represent each mode.

There are two basic elements used in fault tree construction - 'gates' and 'events'. Events can be classified as intermediate or basic. Intermediate events can be developed further and are represented by rectangles in the fault tree diagram. Basic events are represented by circles and cannot be developed any further. Basic events usually are component failures or human errors. These symbols are shown in Table 2.1.

The gates either allow or inhibit the passage of fault logic up through the tree and show the relationships between the 'events' needed for occurrence of a higher event. The development of a fault tree involves use of Boolean expressions represented by logical operations 'Or', 'And', 'Not'. These expressions are represented by gates

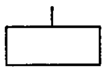
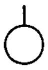
Symbol	Meaning
	Intermediate event further developed by a gate. It indicates that the event is capable of being broken down. This is the only symbol that will have a logic gate and input events below it.
	Basic event. These symbols are found at the bottom of fault trees and require no further development or breakdown.

Table 2.1: Event symbols

'Or', 'And' and 'Not' in a fault tree, respectively. Another gate used in fault tree construction is a 'k out of N' gate, also called 'Vote' gate, which can be expressed as a combination of 'Or' and 'And' gates. This gate allows the flow of logic through the tree if at least k out of N inputs occur. The symbols used to represent these gates are shown in Table 2.2. There are other gates but the ones shown are those most commonly adopted. Before analysis can be performed on any fault tree all gates must be expressed in terms of the 'And', 'Or' and 'Not' gates.

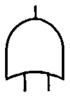
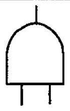
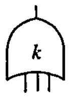
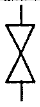
Symbol	Name	Relation
	OR	Output event occurs if at least one of the input events occurs
	AND	Output event occurs if all input events occur
	VOTE	Output event occurs if at least k out of N possible inputs occur
	NOT	Output event occurs if the input event doesn't

Table 2.2: Gate symbols

Once a top event has been determined, it is developed by asking 'what could cause this?'. Hence the immediate, necessary and sufficient causes for its occurrence are determined. In this way, events in the tree are continually redefined in terms of lower resolution events. This process is terminated when basic events are reached.

A system where failure modes are expressed only in terms of component failures is referred to as a 'coherent' system. A coherent fault tree will have only 'Or' and 'And' gates. If failure modes in the system are expressed in terms of component failures and successes, the system is called 'non-coherent'. Non-coherent fault trees also have 'Not' gates.

There can be two types of analysis which can be performed once the fault tree is constructed:

- Qualitative analysis
- Quantitative analysis

2.3 Qualitative Analysis

Qualitative analysis involves the identification of combinations of component states, which cause the system to fail. For coherent fault trees these combinations are called cut sets or minimal cut sets and just involve component failures [7]. In the case of non-coherent fault trees (when 'Not' logic is involved), the combinations of basic events that would cause system failure are called implicants. The minimal sets of implicants are called prime implicants.

The definition of a cut set is:

> A cut set is a collection of basic events whose presence will cause the top event to occur.

System failure, however, does not necessarily need the failure of all the components in a cut set, but for any system the largest cut set will consist of all component failures. Generally only lists of component failures which are necessary and sufficient to cause system failure are looked at. Hence the importance of the minimal cut sets.

> A cut set is said to be minimal if it cannot be further minimized but still insures the occurrence of the top event.
>
> Minimal cut sets are sometimes called the minimal failure modes of a system.

Two fault trees are logically equivalent if they have the same minimal cut sets. The order of a minimal cut set is the number of components within the set. The lowest order minimal cut sets contribute most to system failure, as fewer component failures are needed to cause system failure.

In order to determine the minimal cut sets from a fault tree, Boolean logic expressions for the top event must be transformed to a **sum-of-products** form. This can be achieved using a top-down or bottom-up approach. The top down approach would start with the top event and then gradually substitutes gates with their inputs using Boolean expressions until the expression for the top event consists only of basic events. The bottom-up approach begins at the bottom of the fault tree and works upwards to the top event. Both of these methods are straightforward to apply and involve the expansion of Boolean expressions. The difference between these two approaches is in which end of the fault tree is used to initiate the expansion process. The following laws of Boolean algebra are used to simplify and to remove redundancies in the expressions obtained. In Boolean algebra, '·' is used to represent 'And' and '+' represents 'Or'.

2.3.1 Rules Of Boolean Algebra

1. Commutative laws

   $A + B = B + A$

   $A \cdot B = B \cdot A$

2. Associative laws

   $(A + B) + C = A + (B + C)$

   $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

3. Distributive laws

   $A + (B \cdot C) = (A + B) \cdot (A + C)$

   $A \cdot (B + C) = A \cdot B + A \cdot C$

4. Identities

   $A + 0 = A$

   $A + 1 = 1$

   $A \cdot 0 = 0$

   $A \cdot 1 = 1$

5. Idempotent law

   $A + A = A$

   $A \cdot A = A$

6. Absorption law

   $A + A \cdot B = A$

   $A \cdot (A + B) = A$

7. Complementation

   $A + \overline{A} = 1$

   $A \cdot \overline{A} = 0$

   $\overline{(\overline{A})} = A$

8. De Morgan's laws

   $\overline{(A + B)} = \overline{A} \cdot \overline{B}$

   $\overline{(A \cdot B)} = \overline{A} + \overline{B}$

Laws 5 and 6 enable the removal of redundancies in expressions: law 5 removes repeated cut sets and repeated events within each cut set and law 6 removes non-minimal cut sets.

2.4 *Fault Tree Quantification*

Quantitative analysis of the fault tree allows the calculation of a number of parameters, which are used to assess the system. The top event probability and frequency are used together with the expected number of occurrences of the top event and event importance measures to gain a full understanding of the system.

Quantitative analysis is based on a probabilistic method known as 'Kinetic Tree Theory' introduced by Vesely [8]. The underlying assumption of the Kinetic Tree Theory is that all basic events in the tree structure occur independently of one another.

2.4.1 *Top event probability*

Each system is assumed to exist in one of two states - working or failed. The state of the system will be a function of the state of each component in the system.

Each component is also assumed to exist in one of two states - working or failed. For the ith component the binary indicator variable x_i is defined to be:

$$x_i = \begin{cases} 1 & \text{if component } i \text{ is failed} \\ 0 & \text{if component } i \text{ is working} \end{cases}$$

where $i = 1, 2, \ldots, n$, and n is the number of components in the system.

The system structure function is defined as:

$$\phi(x) = 1 - \prod_{i=1}^{N_C} (1 - \rho_i(x)) = \begin{cases} 1 & \text{if system is failed} \\ 0 & \text{if system is working} \end{cases} \tag{2.1}$$

where $\rho_i(x)$ is the binary indicator function for each minimal cut set $C_i, i = 1 \ldots N_C$:

$$\rho_i(x) = \prod_{j \in C_i} x_j = \begin{cases} 1 & \text{if cut set } C_i \text{ exists} \\ 0 & \text{if cut set } C_i \text{ does not exist} \end{cases} \tag{2.2}$$

The probability of the top event is given by the expected value of the system structure function:

$$Q_{sys}(t) = E[\phi(x)] \tag{2.3}$$

If each minimal cut set is independent (there are no common events between any cut sets), then:

$$\phi[E(x)] = E[\phi(x)] \tag{2.4}$$

Hence the expected value of the structure function for a fault tree without repeated events would be calculated by substituting the probability of failure of each component in the structure function.

However minimal cut sets are not usually independent, and in this case the full expansion of the structure function is needed. For example, if there are two minimal cut sets: $C_1 = \{X_1, X_2\}$, $C_2 = \{X_2, X_3\}$, then the structure function is given by:

$$\begin{aligned} \phi(x) &= 1 - (1 - x_1 \cdot x_2)(1 - x_2 \cdot x_3) \\ &= 1 - (1 - x_1 \cdot x_2 - x_2 \cdot x_3 + x_1 \cdot x_2 \cdot x_2 \cdot x_3) \end{aligned} \tag{2.5}$$

After reduction of the indicator variables (i.e. $X_i = X_i^n$) the following result is obtained:

$$\phi(x) = x_1 \cdot x_2 + x_2 \cdot x_3 - x_1 \cdot x_2 \cdot x_3 \tag{2.6}$$

The probability of the top event T for the fault tree with 2 minimal cut sets given earlier is described by the expected value of the expanded and reduced structure function:

$$\begin{aligned} Q_{sys}(t) &= E[x_1 \cdot x_2 + x_2 \cdot x_3 - x_1 \cdot x_2 \cdot x_3] \\ & P(X_1) \cdot P(X_2) + P(X_2) \cdot P(X_3) - P(X_1) \cdot P(X_2) \cdot P(X_3) \end{aligned} \tag{2.7}$$

An alternative, more efficient way to deal with repeated events is to use Shannon's decomposition formula.

2.4.2 Shannon's decomposition formula

According to Shannon's formula, a Boolean function $f(\underline{x})$, where $\underline{x} = (x_1, \ldots, x_n)$, can be expressed as:

$$f(\underline{x}) = x_i f(1_i, \underline{x}) + \overline{x_i} f(0_i, \underline{x}) \tag{2.8}$$

where $f(1_i, \underline{x})$ represents $f(\underline{x})$ with component x_i failed and $f(0_i, \underline{x})$ represents $f(\underline{x})$ with component x_i working. $f(1_i, \underline{x})$ and $f(0_i, \underline{x})$ are known as the residues of $f(\underline{x})$ with respect to x_i .

The structure function is pivoted around the most repeated variable using Shannon's formula. This is continued until no repeated events are left in the residues.

Applying Shannon's formula to the structure function given in (2.5) and pivoting around variable X_2 gives:

$$\begin{aligned} \phi(x) &= x_2[1-(1-x_1)(1-x_3)] + (1-x_2)[0] \\ &= x_2[1-(1-x_1)(1-x_3)] \end{aligned}$$

The probability of the top event is then given by:

$$Q_{sys}(t) = E[\phi(x)] = P(X_2)[1-(1-P(X_1))(1-P(X_3))] \tag{2.9}$$

An alternative approach to the structure function method to obtain the top event probability is to use the inclusion-exclusion formula.

2.4.3 Inclusion-Exclusion Formula

This approach is suitable whether basic events are repeated or not. The top event T occurs if at least one cut set exists. This gives the following Boolean expression for T :

$$T = C_1 + C_2 + \ldots + C_{N_C} = \bigcup_{i=1}^{N_C} C_i$$

$$P(T) = P\left(\bigcup_{i=1}^{N_C} C_i\right)$$

Expanding this expression gives the inclusion-exclusion expansion:

$$P(T) = \sum_{i=1}^{N_C} P(C_i) - \sum_{i=2}^{N_C}\sum_{j=1}^{i-1} P(C_i \cap C_j) + \ldots + (-1)^{N_C-1} P(C_1 \cap C_2 \cap \ldots \cap C_{N_C}) \tag{2.10}$$

If the number of minimal cut sets, N_i , is large the expression (2.10) becomes tedious and time consuming to calculate. In simulations its calculation may be impractical and hence approximations are used.

2.4.4 Upper and lower bounds for system unavailability

Taking the first two terms of the inclusion-exclusion expansion gives the following:

$$\underbrace{\sum_{i=1}^{N_C} P(C_i) - \sum_{i=2}^{N_C}\sum_{j=1}^{i-1} P(C_i \cap C_j)}_{\text{lower bound}} \le \underbrace{Q_{sys}(t)}_{\text{exact}} \le \underbrace{\sum_{i=1}^{N_C} P(C_i)}_{\text{upper bound}} \tag{2.11}$$

The upper bound of the top event probability is known as the rare event approximation since it is accurate if the component failure events are rare.

2.4.5 Minimal cut set upper bound

A more accurate upper bound is the minimal cut set upper bound.

$$\begin{aligned} Q_{sys}(t) &= P(\text{system failure}) = P(\text{at least 1 minimal cut set occurs}) \\ &= 1 - P(\text{no minimal cut set occurs}) \end{aligned}$$

As

$$P(\text{no minimal cut set occurs}) \ge \prod_{i=1}^{N_C} P(\text{minimum cut set } i \text{ does not occur}),$$

the following is correct:

$$Q_{sys}(t) \leq 1 - \prod_{i=1}^{N_C} (1 - P(C_i)) \tag{2.12}$$

It can be shown that

$$\begin{array}{ccc} Q_{sys}(t) & \leq 1 - \prod_{i=1}^{N_C} (1 - P(C_i)) \leq & \sum_{i=1}^{N_C} P(C_i) \\ \text{exact} & \text{minimal cut set} & \text{rare event} \\ & \text{upper bound} & \text{approximation} \end{array}$$

2.4.6 Top event frequency

The top event frequency or the system failure intensity $w_s(t)$ is defined as the probability that the top event occurs at t per unit time. Therefore $w_s(t)\,dt$ is the probability that the top event occurs in the time interval $[t, t+dt)$.

For the top event to occur between t and $t+dt$ all the minimal cut sets must not exist at t and then one or more minimal cut sets occur during t to $t+dt$. It is assumed that dt is so small that only one component fails in this time. More than one minimal cut set can occur in a small time element dt since component failure events can be common to more than one minimal cut set. This can be expressed as:

$$w_s(t)\,dt = P\left[A \bigcup_{i=1}^{N_c} \theta_i\right] \tag{2.13}$$

where A is the event that all minimal cut sets do not exist at time t and $\bigcup_{i=1}^{N_c} \theta_i$ is the event that one or more minimal cut sets occurs in time t to $t+dt$.

As $P(A) = 1 - P(\overline{A})$, equation 2.13 can be written as:

$$w_s(t)\,dt = P\left[A \bigcup_{i=1}^{N_c} \theta_i\right] = P\left[\bigcup_{i=1}^{N_c} \theta_i\right] - P\left[\overline{A} \bigcup_{i=1}^{N_c} \theta_i\right] \tag{2.14}$$

where $\overline{A}$ means that at least one minimal cut set exists at t.

The first term of the equation 2.14 is the contribution from the occurrence of at least one minimal cut set in the small time element dt and the second term is a correction term representing the contribution of minimal cut sets occurring while other minimal cut sets already exist (i.e. system is already failed). Denoting these terms by $w_s^{(1)}(t)\,dt$ and $w_s^{(2)}(t)\,dt$ respectively gives the following:

$$w_s(t)\,dt = w_s^{(1)}(t)\,dt - w_s^{(2)}(t)\,dt \tag{2.15}$$

Terms on the right side of the equation 2.14 can be expanded using the inclusion-exclusion principle, but as this is computationally intensive approximations may be used.

2.4.7 Approximation of the system unconditional failure intensity

From equation 2.15

$$w_s(t)\,dt \leq w_s^1(t)\,dt \tag{2.16}$$

and hence there is an upper bound $w_{s_{MAX}}(t)$ for $w_s(t)$:

$$w_{s_{MAX}}(t) = w_s^1(t) \tag{2.17}$$

If the component failures are rare events then the minimal cut set failures will also be rare events. The second term of equation 2.15, $w_s^{(2)}(t)\,dt$, requires minimal cut sets to exist and occur at the same time. When component failures are rare this occurrence rate is also very small and hence $w_s(t) \simeq w_{s_{MAX}}(t)$.

As

$$w_s^{(1)}(t)\,dt = \bigcup_{i=1}^{N_C} P(\theta_i) \tag{2.18}$$

results in a series expansion, it can be truncated after the first term to give the rare event approximation:

$$\begin{aligned} w_{s_{MAX}}dt &\leq \sum_{i=1}^{N_C} P(\theta_i) \\ &\leq \sum_{i=1}^{N_C} w_{\theta_i}(t)\,dt \end{aligned} \tag{2.19}$$

where $P(\theta_i)$ is the probability of the occurrence of minimal cut set i; w_{0_i} is the unconditional failure intensity of minimal cut set i.

2.4.8 Expected number of system failures

The expected number of system failures in time t, $W(0,t)$, is given by the integral of the system failure intensity in the interval $[0,t)$:

$$W(0,t) = \int_0^t w_s(u)\,du \tag{2.20}$$

The expected number of system failures is an upper bound for system unreliability:

$$\underset{\text{Unreliability}}{F(t)} \leq \underset{\text{Expected number of system failures}}{W(0,t)}$$

If system failure is rare, this upper bound is a close approximation.

2.5 Example

To illustrate the use of fault tree analysis consider the example shown in Figure 2.1. The top-down approach is demonstrated using this example fault tree.

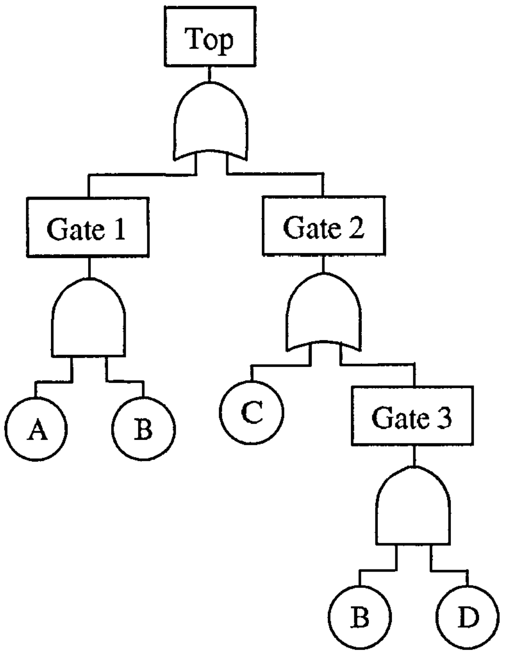


Figure 2.1: Example fault tree

In the top-down approach the starting point is the top event. Then it is expanded by substituting each gate in the expression by events appearing lower down in the fault tree and simplifying the expression until it has only basic component failures. The top event in Figure 2.1 has an 'Or' gate with two inputs:

$$Top = Gate1 + Gate2$$

Gate 1 is an 'And' gate with two input events, A and B:

$$\begin{aligned} Gate1 &= A \cdot B \\ Top &= A \cdot B + Gate2 \end{aligned}$$

Gate 2 is an 'Or' gate with two inputs, C and $Gate3$:

$$\begin{aligned} Gate2 &= C + Gate3 \\ Top &= A \cdot B + C + Gate3 \end{aligned}$$

Gate 3 is an 'And' gate with two input events, B and D:

$$Gate3 = B \cdot D$$

Hence, the following expression for the Top event is obtained:

$$Top = A \cdot B + C + B \cdot D$$

This is the minimal disjunctive form of the logic equation, each term of which is a minimal cut set. This fault tree therefore has three minimal cut sets, one of order one and two of order two: $\{C\}, \{A, B\}, \{B, D\}$.

The probability of the top event using the inclusion-exclusion would be calculated as follows:

$$\begin{aligned} P(Top) &= P(A \cdot B) + P(C) + P(B \cdot D) \\ &\quad - P(A \cdot B \cdot C) - P(A \cdot B \cdot D) - P(C \cdot B \cdot D) \\ &\quad + P(A \cdot B \cdot C \cdot D) \end{aligned}$$

Minimal cut set bound (see equation 2.12) for this system would be:

$$P(Top) \leq 1 - (1 - P(C))(1 - P(A \cdot B))(1 - P(B \cdot D))$$

The rare event approximation (equation 2.11) is:

$$P(Top) \leq P(C) + P(A \cdot B) + P(B \cdot D)$$

2.6 Importance measures

An importance analysis is a sensitivity analysis which identifies weak areas of the system and can be very valuable at the design stage. For each component its importance measure signifies the role that it plays in either causing or contributing to the occurrence of the top event. This allows components or cut sets to be ranked according to the extent of their contribution to the occurrence of the top event.

Importance measures can be categorised as deterministic or probabilistic. Probabilistic measures can also be categorised into those dealing with system availability assessment and those concerned with system reliability assessment.

2.6.1 Deterministic measures

Deterministic measures assess the importance of a component to the system operation without considering the component's probability of occurrence. One such measure is the structural measure of importance.

2.6.1.1 Structural measure of importance

The structural measure of importance for a component i is defined by equation 2.21:

$$I_i^{ST} = \frac{\text{number of critical system states for component } i}{\text{total number of states for the } (n-1) \text{ remaining components}} \tag{2.21}$$

A system state for component i will be described as a critical state if failure of component i causes the system to go from a working to a failed state.

2.6.2 Probabilistic measures (System Availability)

Probabilistic measures are generally of more use than deterministic measures in reliability problems as they take into account the component's probability of failure.

2.6.2.1 Birnbaum's measure of importance

Birnbaum's measure of importance is also known as the criticality function. The criticality function for a component i, $G_i(\mathbf{q}(t))$, is defined as the probability that the system is in a critical system state for component i.

The two expressions for the criticality function are:

1.

$$G_i\left(\mathbf{q}\left(t\right)\right) = Q\left(1_i, \mathbf{q}\left(t\right)\right) - Q\left(0_i, \mathbf{q}\left(t\right)\right) \tag{2.22}$$

where $Q\left(t\right)$ is the probability that the system fails, $\left(1_i, \mathbf{q}\right) = \left(q_1, \ldots, q_{i-1}, 1, q_{i+1}, \ldots, q_n\right)$, $\left(0_i, \mathbf{q}\right) = \left(q_1, \ldots, q_{i-1}, 0, q_{i+1}, \ldots, q_n\right)$.

This expression gives the probability that the system fails with component i failed minus the probability that the system fails with component i working. So, this gives the probability that the system fails only if component i fails.

2.

$$G_i\left(\mathbf{q}\left(t\right)\right) = \frac{\partial Q\left(\mathbf{q}\right)}{\partial q_i} \tag{2.23}$$

This defines the criticality function as a partial derivative which is the same as the first expression 2.22 as:

$$\frac{\partial Q\left(\mathbf{q}\right)}{\partial q_i} = \frac{Q\left(1_i, \mathbf{q}\left(t\right)\right) - Q\left(0_i, \mathbf{q}\left(t\right)\right)}{1 - 0} \tag{2.24}$$

2.6.2.2 Criticality measure of importance

The criticality measure of importance is defined as the probability that the system is in a critical state for component i, and i has failed (weighted by the system unavailability Q_{sys}):

$$I_i^{CM} = \frac{G_i\left(\mathbf{q}\left(t\right)\right) q_i\left(t\right)}{Q_{sys}\left(\mathbf{q}\left(t\right)\right)} \tag{2.25}$$

2.6.2.3 Fussell-Vesely measure of importance

This measure of importance is defined as the probability of union of the minimal cut sets containing component i given that the system has failed:

$$I_i^{FV} = \frac{P\left(\bigcup_{k|i\in k} C_k\right)}{Q_{sys}\left(\mathbf{q}\left(t\right)\right)} \tag{2.26}$$

The importance rankings by Fussell-Vesely method are very similar to those produced by the criticality measure of importance (2.25).

2.6.2.4 *Fussell-Vesely measure of minimal cut set importance*

This measure provides a similar function to the previously defined importance measures for components except that the minimal cut sets are themselves ranked. The importance measure is defined as the probability of occurrence of cut set i given that the system has failed:

$$I_{C_i}^{FV} = \frac{P(C_i)}{Q_{sys}(\mathrm{q}(t))} \tag{2.27}$$

2.6.3 *Probabilistic measures (Systems reliability)*

Probabilistic measures for system reliability are appropriate for systems where the interval reliability is being assessed and the sequence in which components fail matters. The sequence of failure can be described with the use of enabling and initiating events. This is of particular use when analysing safety protection systems. For example, if a hazardous event occurs after the protection system failed, this would result in a dangerous system failure. However, if the protection system was working when the hazardous event occurred, but failed later, then it would shutdown the system and a dangerous situation would be avoided. So, in this example, the hazardous event is an initiator, as it would result in a system failure only if the enabling event has already occurred. If the initiating event occurs first, then the safety system would respond as required and danger would be avoided. Initiating and enabling events are defined as follows:

> Initiating events perturb system variables and place a demand on control/protective systems to respond.
>
> Enabling events are inactive control/protective systems which permit initiating events to cause the top event.

All probabilistic measures for system reliability are weighted according to the expected number of system failures, $W(0,t)$.

2.6.3.1 *Barlow-Proschan measure of initiator importance*

The Barlow-Proschan measure of initiator importance is the probability that the initiating event i causes the system failure over the interval $[0,t)$. It is defined in terms of the criticality function and the unconditional failure intensity of the component:

$$I_i^{BP} = \frac{\int_0^t \{Q(1_i, \mathbf{q}(t)) - Q(0_i, \mathbf{q}(t))\} w_i(t)\, dt}{W(0,t)} \tag{2.28}$$

2.6.3.2 Sequential contributory measure of enabler importance

The sequential contributory measure of enabler importance is the probability that enabling event i permits an initiating event to cause system failure over the interval $[0,t)$. The failure of the enabler i is only a factor when it is contained in the same minimal cut set as the initiating event j:

$$I_e^{BP} = \frac{\sum\limits_{\substack{j \\ i \neq j \\ i \text{ and } j \in C_k \\ \text{for some } k}} \int_0^t \{Q(1_i, 1, j, \mathbf{q}(t)) - Q(1_i, 0_j, \mathbf{q}(t))\} q_i(t)\, w_j(t)\, dt}{W(0,t)} \tag{2.29}$$

This expression is only an approximation.

2.6.3.3 Barlow-Proschan measure of minimal cut set importance

This measure of cut set importance is the probability that a minimal cut set i causes the system failure in interval $[0,t)$ given that the system has failed:

$$I_i = \frac{\sum\limits_{j \in i} \int_0^t \left[1 - Q\left(0_j, 1^{i-\{j\}}, \mathbf{q}(t')\right)\right] \prod\limits_{\substack{k \neq j \\ k \in i}} q_k(t')\, w_j(t')\, dt'}{W(0,t)} \tag{2.30}$$

j is each initiating event in the minimal cut set $\{i\}$.

2.7 Summary

Fault tree analysis is very important and frequently used to quantify system performance. It gives a diagrammatic representation of the system failure causes, and also provides a means for system quantification. Performing analysis upon large fault trees (quantitative or qualitative) may be computationally intensive and hence approximations are needed for some parameters and that will lead to loss of accuracy.

3. BINARY DECISION DIAGRAMS

3.1 Introduction

Fault trees described in the previous chapter are a good way to represent the logic of the system. However, if the fault tree is large, then performing analysis on it can be computationally expensive. Approximations are needed for many parameters and that would result in loss of accuracy. A more accurate and efficient way to perform these calculations is to use the Binary Decision Diagram technique.

Binary Decision Diagrams (BDDs) were introduced by Lee [9] who used them to represent switching circuits. They were further studied by Akers [10] who defined a digital function in terms of a diagram, which told the user the output value of the function by examining the values of its inputs. The BDDs were first applied to reliability and, more specifically, to fault tree analysis, in 1980's by Schneeweiss [11]. Further development of the use of BDDs in reliability analysis was developed by Rauzy [12], who suggested that they could provide an alternative technique for performing fault tree analysis.

The BDD method first converts a fault tree to a binary decision diagram which can then be used for analysis. In order to do this, an order in which components are considered must be taken. The BDD represents the Boolean equation for the top event, which is much easier to analyse than a fault tree. The method allows for quantitative and qualitative analysis of the fault tree. The advantage of this method compared to fault tree analysis is that exact solutions can be calculated efficiently without the need for approximations.

3.2 Description of the BDD

A BDD is a directed acyclic graph. According to Rauzy[12], BDDs have two important features:

- the graphs are compacted by sharing equivalent subgraphs;
- the results of operations performed on BDD are memorised and thus a job is never performed twice.

A BDD is composed of terminal and non-terminal nodes (vertices), connected by branches. The non-terminal nodes encode basic events and the terminal nodes correspond to the final state of the system. The example of a BDD is shown in Figure 3.1.

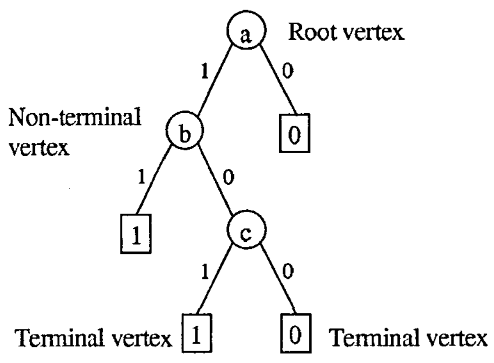


Figure 3.1: Example of Binary Decision Diagram

A non-terminal node of a BDD has two outgoing branches: if the basic event represented by the non-terminal node occurs, then the diagram is further developed following the left-hand side branch ('1' branch), and if the basic event doesn't occur the diagram is developed on the right hand side branch ('0' branch). In the following work, all left branches of a BDD will represent '1' branches and all right branches will represent '0' branches. The size of the BDD is usually measured by the number of non-terminal nodes. Terminal nodes have the value 1 if the top event occurs (i.e. system fails) or 0 if the top event doesn't occur (i.e. system doesn't fail).

All paths through the diagram start at the root vertex, the top node, and proceed to a terminal node marking the end of the path. A path terminating in node '1' gives a cut set of the fault tree. Only nodes lying on the '1' branches of the path are included in the cut set.

3.3 Construction of the BDD

3.3.1 Construction of the BDD using structure function

One method to construct a BDD from a fault tree is to use the structure function $\phi(\underline{x})$ of the system. An order in which components will be considered in the construction process is important as it can significantly influence the size of BDD. Once an order of components is determined, values of 1 and 0 are substituted for each component in the structure function according to the chosen ordering. To illustrate the process a fault tree shown in Figure 3.2 is used.

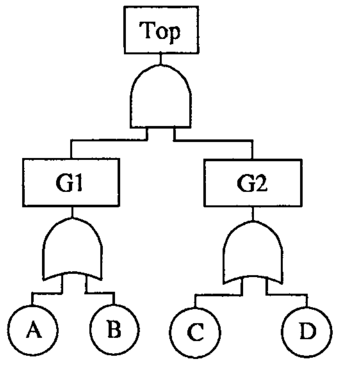


Figure 3.2: Fault Tree Example

This fault tree has four minimal cut sets:

1. $\{A, C\}$
2. $\{A, D\}$
3. $\{B, C\}$
4. $\{B, D\}$

which gives the following structure function:

$$\phi(\underline{x}) = 1 - (1 - x_A \cdot x_C)(1 - x_A \cdot x_D)(1 - x_B \cdot x_C)(1 - x_B \cdot x_D) \tag{3.1}$$

Using top-down, left-right ordering scheme (simply ordering the variables as they are encountered on a top-down, left-right traversal of the fault tree) the component order would be:

$$A < B < C < D$$

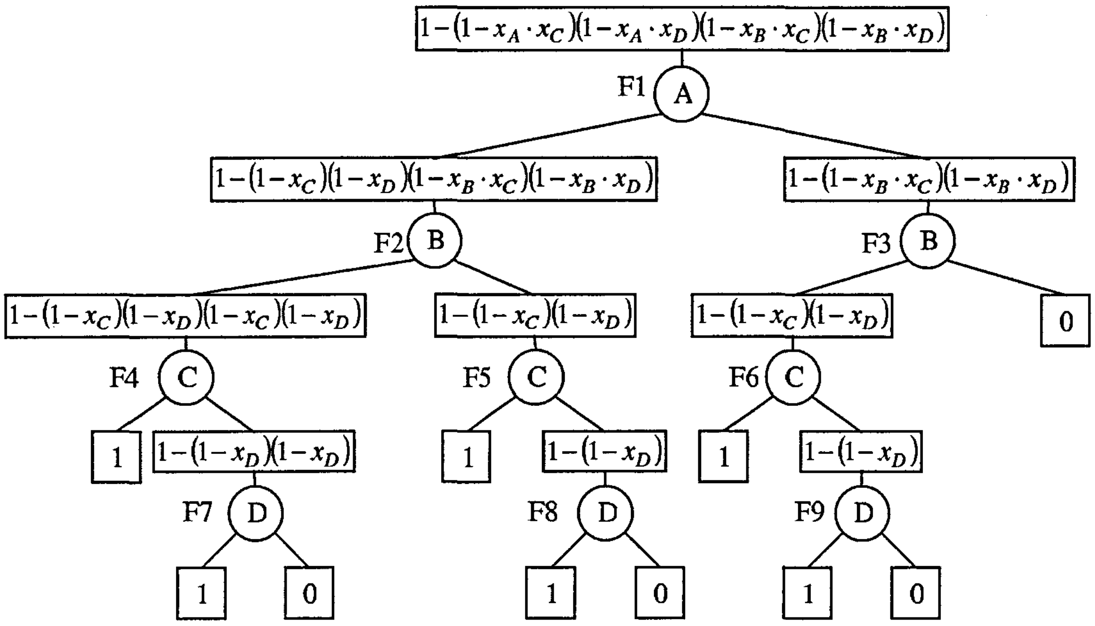


Figure 3.3: Binary Decision Diagram for Fault Tree shown in Figure 3.2

This means that basic event A is considered first, then basic event B, then C and finally basic event D. The first node (root vertex) represents basic event A. The result of the left-hand branch is obtained by substituting the value 1 into the structure function for each x_A and the result for the right-hand side branch is obtained by substituting value 0 for A:

$$\begin{aligned} x_A = 1 : \quad & \phi(\underline{x}) = 1 - (1 - x_C)(1 - x_D)(1 - x_B \cdot x_C)(1 - x_B \cdot x_D) \quad (3.2) \\ x_A = 0 : \quad & \phi(\underline{x}) = 1 - (1 - x_B \cdot x_C)(1 - x_B \cdot x_D) \end{aligned}$$

Other basic events are considered in the same way until the terminal nodes are reached. The resulting BDD is shown in Figure 3.3.

The resulting BDD is not in its most efficient form and although it will generate cut sets, these are not minimal. A BDD can be made more efficient by applying collapsing operations. These can be applied to equivalent nodes where, from Friedman and Supowit [13], two nodes of a BDD are equivalent if they both are:

- terminal nodes with the same value, or
- non-terminal nodes having the same label and their left sons are equivalent and their right sons are equivalent.

The son of a node is the node to which either the '1' or '0' branch leads.

The following 'collapsing' operations can be used to reduce the size of a BDD:

1. If two sons of node A are equivalent, then delete node A and direct all of its incoming branches to its left son.
2. If nodes A and B are equivalent, then delete node B and direct all of its incoming branches to A.

The above operations can be used to reduce the BDD shown in Figure 3.3. Operation 1 can be applied to node F2 as both its sons are equivalent. This results in the incoming branch from node F1 being directed to the left son of F2, node F4. Therefore, nodes F2, F5 and F8 are deleted. Then operation 2 can be applied to equivalent nodes F4 and F6. Following the rule, node F6 is deleted and the incoming branch from node F3 is directed to node F4. The resulting BDD is shown in Figure 3.4.

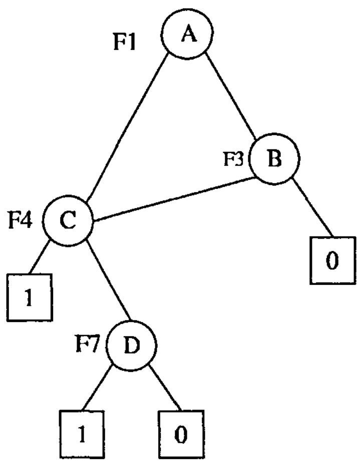


Figure 3.4: Reduced BDD from Figure 3.3

The reduced BDD is much smaller than the original. It has four non-terminal nodes compared with nine in the original. It must be noted, that this reduction does not change the logic of the BDD.

3.3.2 Construction Of the BDD Using If-Then-Else Approach

The if-then-else (*ite*) method for constructing BDD's was developed by Rauzy [12]. It is derived from Shannon's formula given in equation 2.8:

$$f\left(\underline{x}\right) = x_1 f_1 + \overline{x_1} f_2 \tag{3.3}$$

where f_1 represents $f\left(\underline{x}\right)$ with $x_1 = 1$ and f_2 represents $f\left(\underline{x}\right)$ with $x_1 = 0$. Functions f_1 and f_2 are one order less than $f\left(\underline{x}\right)$.

Each non-terminal node in the BDD has an *ite* structure of the form $ite\left(x_1, f_1, f_2\right)$ where x_1 is a Boolean variable and f_1 and f_2 are logic functions. This means: if x_1 fails then consider f_1 else consider f_2. In the BDD structure f_1 would be at the end of the '1' branch of the node x_1 and f_2 would be at the end of '0' branch. The structure is represented in Figure 3.5.

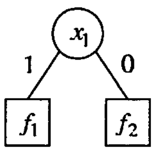


Figure 3.5: *ite* structure for component x_1

Variable ordering must be chosen before construction of the BDD. Then each basic event x_i is assigned the *ite* structure $ite\left(x_i, 1, 0\right)$. The following rules are then used for manipulation of *ite* structures:

If $J = ite\left(x, f_1, f_2\right)$ and $H = ite\left(y, g_1, g_2\right)$, then

1. $x < y$ (x appears before y in the variable ordering)

$$J * H = ite\left(x, f_1 * H, f_2 * H\right) \tag{3.4}$$

2. $x = y$

$$J * H = ite\left(x, f_1 * g_1, f_2 * g_2\right) \tag{3.5}$$

where $*$ corresponds to a Boolean operation 'And' or 'Or'.

To simplify the results the following properties are also used:

$$\begin{array}{ll} 1 + H = 1 & 1 \cdot H = H \\ 0 + H = H & 0 \cdot H = 0 \end{array} \tag{3.6}$$

To illustrate the method the fault tree shown in Figure 3.6 is considered.

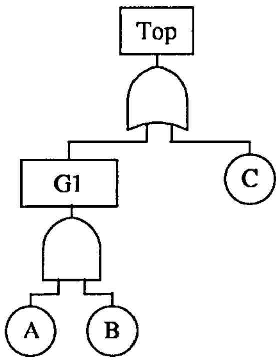


Figure 3.6: Fault Tree Example

Using the top-down left-right ordering strategy the variable order is $C < A < B$. The *ite* structures for each basic event are:

$$A = ite(A, 1, 0)$$
$$B = ite(B, 1, 0)$$
$$C = ite(C, 1, 0)$$

Gate G1 can be expressed (using rule 1 in (3.4) and (3.6)) as :

$$\begin{aligned} \mathrm{G1} &= A \cdot B = ite(A, 1, 0) \cdot ite(B, 1, 0) = \\ &= ite(A, ite(B, 1, 0), 0) \end{aligned}$$

The *ite* structure for the event Top is given (using rule 1 (3.4) and (3.6)) by:

$$\begin{aligned} \mathrm{Top} &= C + \mathrm{G1} = ite(C, 1, 0) + ite(A, ite(B, 1, 0), 0) = \\ &= ite(C, 1, ite(A, ite(B, 1, 0), 0)) \end{aligned} \tag{3.7}$$

To construct the BDD from 3.7 '1' and '0' branches are considered for each variable in turn. For example, C is the first basic event in the variable ordering and it is encoded in the root node of the BDD structure. At the end of '1' branch is a terminal node 1 and the structure $ite(A, ite(B, 1, 0), 0)$ is at the end of '0' branch. Basic event A is considered next and is encoded in the node at the end of the right-hand branch ('0' branch). Its left-hand branch ('1' branch) will end in the structure $ite(B, 1, 0)$, while its right-hand side ('0') branch will terminate in a terminal node 0. The process is repeated once more for the basic event B. The resulting BDD is shown in Figure 3.7.

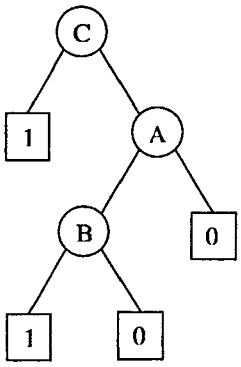


Figure 3.7: Binary Decision Diagram for Fault Tree shown in Figure 3.6

3.4 Qualitative Analysis Of the BDD

Each path from the root node of a BDD to a terminal node '1' defines a solution of the Boolean function $f(\underline{x})$ [12]. Only nodes lying on the '1' branches of the path are included in the cut set. For the BDD shown in Figure 3.7 the cut sets are:

1. $\{C\}$
2. $\{A, B\}$

These are also minimal cut sets. But the BDD does not always produce a list of minimal cut sets. To obtain minimal cut sets the BDD can be minimised or the list of cut sets can be reduced using Boolean algebra rules.

3.4.1 Minimisation

Only if a BDD is in its minimal form will the cut sets produced from it be minimal. A minimisation process for a BDD, developed by Rauzy [12], is applied to its *ite* form and creates a new BDD which defines all minimal cut sets of the fault tree. All shared nodes must be expanded before minimisation.

Let f be a Boolean function of the BDD. If σ is a solution of f, then a path exists from the root of the BDD to terminal node '1' which defines a solution δ of f such that δ is included in σ.

Consider any node in the BDD, the output of which is represented by the function F where:

$$F = ite\,(x, G, H)$$

If δ is a minimal solution of G, then the intersection of $\{\delta\} \cap x$ is a solution of F. In addition, if δ is not a minimal solution of H, then a solution of F smaller than $\{\delta\} \cap x$ does not exist and $\{\delta\} \cap x$ is minimal. The set of all minimal solutions of F will also include minimal solutions of H:

$$sol_{\min}(F) = \{\sigma\} \tag{3.8}$$

$$\sigma = [\{\delta\} \cap x] \cup [sol_{min}(H)] \tag{3.9}$$

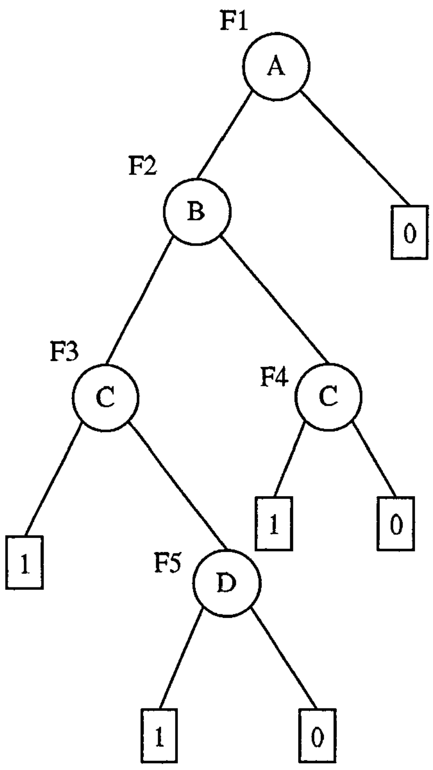


Figure 3.8: Example BDD for minimisation

This algorithm can be applied to the BDD in Figure 3.8. This BDD would produce these cut sets:

1. $\{A, B, C\}$
2. $\{A, C\}$
3. $\{A, B, D\}$

As this BDD is not minimal, it does not generate minimal cut sets. The first cut set is redundant as it contains the second cut set as a subset. To minimize the BDD, each node is considered in turn:

$F1 = ite\,(A, F2, 0)$ - F2 does not contain any paths that are included in '0' branch, as this leads to terminal vertex.

$F2 = ite\,(B, F3, F4)$ - Event 'C' is included in a path on both the '1' branch ($F3$) and the '0' branch ($F4$). Therefore, 'C' is removed from the '1' branch as this will be a non-terminal son of $F2$. This is done by replacing the terminal '1' vertex with a terminal '0' vertex.

$F3 = ite\,(C, 1, F5)$ - $F5$ does not contain any paths that are included in the '1' branch as it leads to the terminal vertex.

$F4 = ite\,(C, 1, 0)$ - Both the '1' and '0' branches are terminal.

$F4 = ite\,(D, 1, 0)$ - Both the '1' and '0' branches are terminal.

The minimised BDD is shown in Figure 3.9.

This BDD produces the following minimal cut sets:

1. $\{A, C\}$
2. $\{A, B, D\}$

The minimised BDD didn't produce the redundant minimal cut set $\{A, B, C\}$.

This technique can only be used to obtain the minimal cut sets as it destroys the structure function form of the BDD and hence the minimised BDD must not be used for quantification.

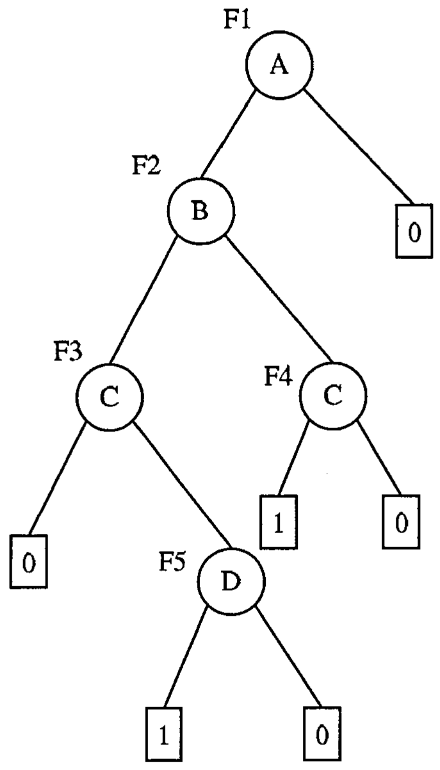


Figure 3.9: The minimised BDD

3.5 Quantitative Analysis Of the BDD

The probability of the root event can be expressed by the BDD as the sum of probabilities of the paths that lead from the root node to any terminal node '1' as these paths will give minimal cut sets. For quantitative analysis nodes lying on the '0' branches of the path are included as well. For the component i that lies on '0' branch the probability of occurrence is described as $\overline{q}_i, \overline{q}_i = 1 - q_i$. For the BDD shown in Figure 3.7 the paths to consider would be

1. C
2. $\overline{C}AB$

Therefore the probability of top event occurrence would be:

$$Q_{TOP} = q_C + (1 - q_C)\, q_A q_B$$

3.6 Summary

The BDD technique is useful to identify the minimal cut sets of a fault tree and to calculate the exact probability of the top event. The difficulty with the technique is the conversion of a fault tree to a BDD as variable ordering can significantly influence the size of the resulting BDD. However, for large systems the BDD method allows more accurate analysis than is possible to achieve using traditional methods, i.e. fault tree analysis.

4. CAUSE-CONSEQUENCE DIAGRAMS

4.1 Introduction

The purpose of risk analysis is to assess probabilities of accidents and evaluate their consequences. Techniques adopted (such as fault tree analysis, Markov analysis, etc.) are incapable of identifying all possible causes and consequences of a critical event.

The cause-consequence diagram method, which was developed at RISO Laboratories, Denmark, by Nielsen [5] in 1971, is a method which presents logical connections between causes of an undesired (critical) event and the consequences of such an event, if one or more preventing/limiting provisions fail [14, 15]. It was initially developed as a graphical tool for analysing relevant accidents in a complex nuclear power plant. It has subsequently been applied to various industrial systems [16]. EDF (Electricite de France) also applied the method to the reliability study of safety-related systems in nuclear power plants and the method was found to be advantageous to other methods previously adopted, essentially for certain mechanical systems [17].

In developing the methods Nielsen noticed that a given accident may be characterised by a 'cause', a sequence of events where the time between the occurrence of the single event can be an important parameter, and finally by the consequences of the accident, when the method should be able to determine all possible causes and consequences that some critical event may lead to if one or more limiting provisions fail. Nielsen [5] states that the method should also provide a basis for determination of the probabilities of any single consequence.

The principle difference between fault trees and cause-consequence diagrams is that the cause-consequence diagram retains information about the order in which the components in the system are called upon [18] and is able to model not only causes of system failure, but also consequences. Event trees are usually used to map the developments from the initiating event to the set of all possible outcomes, but not to determine causes of the failure. By combining both causes and consequences of the critical event, the cause-consequence diagrams also provide the way for easy quantification as the logic is very similar to binary decision diagrams. Nielsen and others [19] noted, that compared with the event trees the cause-consequence diagram gives a better representation of event sequences and the conditions under which these events can take place. The cause-consequence diagram has a benefit of the use of simple, comprehensible symbols that facilitate the communication between different people in the development and commissioning of the system.

4.2 Cause-Consequence Diagram Method

The main principle of the cause-consequence diagram technique is based on the occurrence of a critical event, which for example may be an event involving the failure of components or subsystems, that is likely to produce undesired consequences. Once a critical event has been identified, all relevant causes of it and its potential consequences are developed using two conventional reliability analysis methods - fault tree analysis and event tree analysis [6].

The 'cause' part of the diagram (cause searching) is a fault tree. Fault tree analysis is used to describe the causes of an undesired event. The construction of the tree begins with the definition of the top event (the critical event). Then the causes are indicated and connected with the top event using logical gates 'And' and 'Or' and this procedure is iterated until all causes are fully developed.

The 'consequence' part of the diagram (consequence searching) is an event tree (event-sequential diagram) showing the consequences that a critical event may lead to if one or more preventing/limiting systems do not function as supposed. The event tree method is used to identify the various paths that the system could take, following the critical event, depending on whether certain subsystems or components function correctly or not.

With a combination of fault tree, representing causes of the critical event, and event tree, listing all possible consequences, the logical connection between the causes of a critical event and its consequences can be established. Compared with fault tree analysis, the cause-consequence diagram method gives a simpler representation of event sequences and the conditions under which these events can take place [19].

The relationship between the two reliability methods is shown in Figure 4.1.

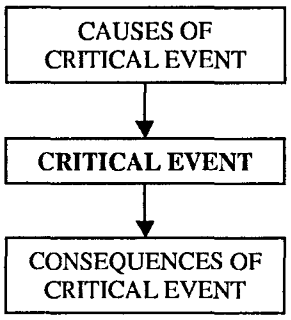


Figure 4.1: Cause-consequence diagram structure

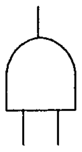
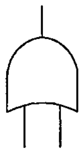
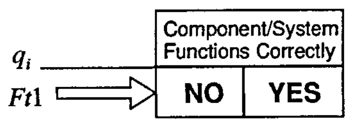
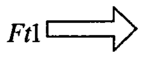
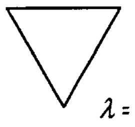
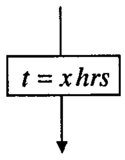
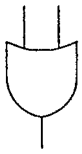
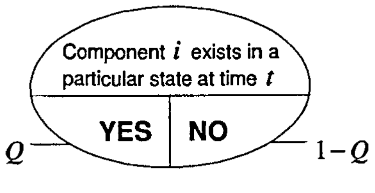
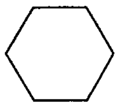
4.3 Symbols for the cause-consequence diagram

The symbols for construction of the cause-consequence diagram are listed in Table 4.1. The symbols for the cause part are the same as those used for the fault tree method. For the consequence part new symbols were developed [5, 17, 6].

The main symbol used in the construction of the consequence diagram is the decision box. The decision box was proposed by Nielsen and is an identical representation of 'YES - NO' branches of an event tree structure. The connection point between the cause and consequence diagrams is the NO branch of the decision boxes as the failure causes of the system, represented by a decision box, are developed using fault tree analysis. Nielsen notes the importance of the delay symbol. The delay symbol is used in constructing consequence diagrams for systems where time delay is important as the knowledge of this may help the analyst to differentiate the different outcomes of the system.

To illustrate a typical cause-consequence diagram the simple system for lighting a lamp can be used (Figure 4.2) [17, 22]. A cause-consequence diagram for this system is represented in Figure 4.3. The initiating (critical) event is 'operator depresses button'. The causes why the bulb is not alight can be that the battery fails to produce power (BAT), the bulb has blown (B) or the fuse is broken (F). Two consequences are considered: there is no light (NL) or the bulb is alight (L).

Table 4.1: Symbols used for the cause-consequence diagram

Symbols for the cause diagram	
	AND gate allows the causality to pass up the tree if at any time all inputs to the gate occur
	OR gate allows causality to pass up through the tree if at any time at least one input to the gate occurs
Symbols for the consequence diagram	
 Component/System Functions Correctly q_i $Ft1$ NO YES	The decision box represents the functionality of a component/system. The NO box represents failure to perform correctly, the probability of which is obtained via a fault tree or single component probability q_i
	Fault tree arrow represents the number of the fault tree structure that corresponds to the decision box
	The initiator triangle represents the initiating event for a sequence where λ indicates the rate of occurrence
	Time delay indicates that the time starts from the time at which the delay symbol is entered and continues up to the end of the time interval in the delay symbol
	OR gate symbol is used to simplify the cause-consequence diagram when more than one decision box enters the same decision box or consequence box
 Component i exists in a particular state at time t YES NO Q $1-Q$	The existence box represents a component existing in a certain state
	The consequence box represents the outcome event due to a particular sequence of events

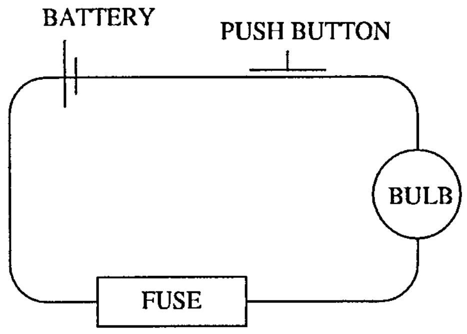


Figure 4.2: Simple light circuit

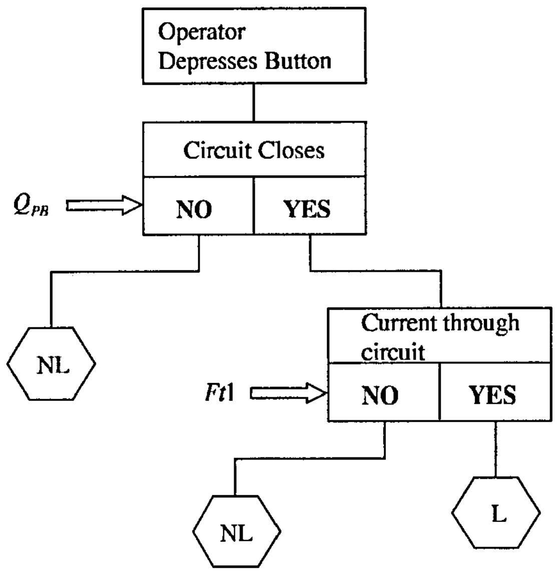


Figure 4.3: Cause-consequence diagram for the light circuit

The light switch circuit functions when an operator depresses the push button, which sends power to the bulb. Once the critical event is identified, the next stage in the construction process is to identify all possible consequences. Following the initiating event the circuit should close causing a current to be applied to the bulb. The cause-consequence diagram is completed by considering the functionality of the components that control the closure of the circuit and the current through the circuit.

The causes of the circuit failing to close is that the push button fails to close the circuit. Therefore a single probability that the push button fails to close, Q_{PB}, is attached to the NO outlet branch of the first decision box. The causes of the circuit

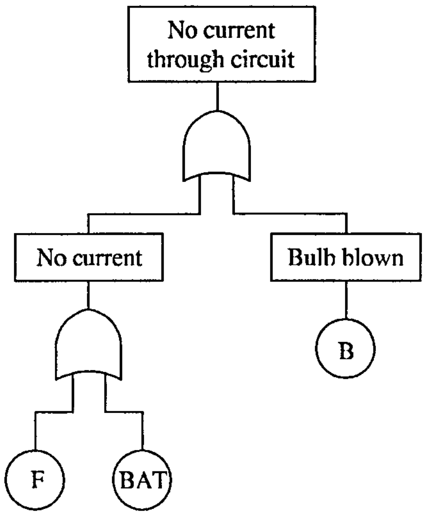


Figure 4.4: Fault tree FT1 for the cause-consequence diagram shown in figure 4.3

failing are that the battery fails to produce power (BAT), the bulb has blown (B), or the fuse is broken (F). These failure causes are shown in Figure 4.4.

Quantification of the cause-consequence diagram, for a system containing only independent failures, can be evaluated by multiplying probabilities of each outlet branch leading to a consequence. The overall probability for any particular consequence is obtained by summing all sequence probabilities that lead to that particular consequence. For example, the probability of light failure, 'NL', in Figure 4.3 is equal to $Q_{PB} + (1 - Q_{PB})Q_{Ft1}$, where Q_{Ft1} is the probability that there is no current through circuit.

4.4 Construction rules

Nielsen [5] gives descriptions of the rules for constructing the cause-consequence diagram. This method can be divided into two main groups that may be called:

1. The cause diagram method (cause searching)

2. The consequence diagram method (consequence searching)

4.4.1 The cause diagram method

The cause diagram is a fault tree relating events and conditions to a particular undesired event which might be, for instance, a relevant system failure. Only events that might contribute to the undesired event should be considered.

The method is characterized by the following points [5, 6]:

1. *Identification of the top event.* The construction of a cause diagram (fault tree) begins with the exact definition of the critical event. Nielsen describes a critical event [15] as an unintended function of a component that controls or effects main energy or mass balance, which can lead to significant consequences. He suggests that it may be expedient to choose a radical abnormal change of a process parameter (e.g. feed water flow stops) or a process variable that exceeds a safety limit (e.g. pressure exceeds trip pressure). The description of the critical event may vary depending on the system considered.

2. *Cause diagram development.* Using a deductive process, the causes of the undesired event are discovered and connected by means of logical gates. The procedure is repeated until all events have been fully developed, i.e. the branches terminate in basic events.

3. *Validation of the diagram.* For each gate used in the diagram the input events must always be both necessary and sufficient, in the context of the gate, to produce the output event.

Generally, at the development of a cause diagram, special attention should be directed towards identification of common mode failures, i.e. simultaneous failures of two or more functionally independent system parts from a common cause [5].

4.4.2 The consequence diagram method

The consequence diagram is a graphical method showing the consequences that the critical event may result in. This method can also be useful for the determination of the probability of each consequence. The construction of the consequence diagram starts with definition of the critical event and following sequences of events, consequences are determined.

The principle of the method is that the starting point is the definition of a critical event, and the objective is to describe how all possible consequences may occur depending on how other systems respond to the critical event.

The consequence diagram method is then constructed by the following methodology:

1. *Component ordering.* The first step of the consequence diagram construction is deciding on the order in which component functioning/failure events are to be taken. To ensure a logical development of the causes of the system failure mode, it was decided that the ordering should follow the temporal action of the system, for example the system activation for the function required given an initial critical event.

2. *Consequence diagram development.* The second stage involves the actual construction of the diagram. Starting from the initiating component, the functionality of each component or subsystem is investigated and the consequences of these sequences determined. If the decision box is governed by a subsystem, then the probability of failure will be obtained via a fault tree diagram.

3. *Reduction.* If any decision boxes are deemed irrelevant, for example the boxes attached to the NO and YES branches are identical and their outcomes and consequences are the same, then these should be removed and the diagram reduced to a minimal form. Removal of these boxes will in no way affect the end result.

An example of the construction of the cause-consequence diagram is given in Section 4.6.

Devised rules for the correct construction of the cause-consequence diagram for a static system are given by Andrews and Ridley [23].

4.4.3 Rules for dependent failure events

The procedure for analysis of an independent system[1] modelled using a cause-consequence diagram begins with the assignment of probabilities to each outlet branch stemming from a decision box. Following this, the probability of any one sequence is obtained by multiplication of the probabilities associated with each decision box [18]. The probability of any particular consequence is then obtained by the summation of probability of each sequence that terminates in that consequence. This procedure cannot be employed unless the failures of each decision box in a sequence are independent. Dependencies may exist in the cause-consequence

[1]Independent system is a system where all components perform independently

diagram, and these must be dealt with before the quantification of the diagram. Andrews and Ridley [6] give the guidelines on how to deal with dependent failure events (common failure and inconsistent failure events).

4.4.3.1 Common failure events

Andrews and Ridley [6] noticed that the first type of dependency that may arise is that the same failure event exists in more than one fault tree structure on the same path in the cause-consequence diagram. In order to deal with the common failure event, the event is extracted from the fault tree structure and placed in a new decision box preceding the first decision box that contains the common failure event. The original cause-consequence diagram is then duplicated on each outlet branch stemming from the new decision box. Following the NO outlet branch of the new decision box, the failure event is set to TRUE in any fault tree structure in which it is found. Similarly, following the YES outlet branch, the probability of failure of the common failure event is set to FALSE in any fault tree structure in which it is present.

4.4.3.2 Inconsistent failure events

As Andrews and Ridley [6] note, in certain systems components are required to perform different functions which, if successfully accomplished, result in the components residing in different states at different times. For example, initially a relay may be required to be closed and later in the sequence be open.

The simple cause-consequence diagram section shown in Figure 4.5 can be used with corresponding fault trees depicted in Figure 4.6. K2 is a relay that can fail closed (K2FC) or open (K2FO). To start the motor relay K2 is required to close. It may fail open because of relay failure or some operational failure P1. If K2 closes as required, motor should start. Motor will fail to start if there is some problem with the motor or some other operational problems (P3). Once the motor starts relay K2 is required to open. If K2 is failed closed or there is some operational failure (P2), it will fail to open. If K2 contacts do not open, then system fails. If K2 contacts open, the system starts to operate.

For systems that are not in continuous operation, certain component failures could occur between operations. For example, the relay could fail between

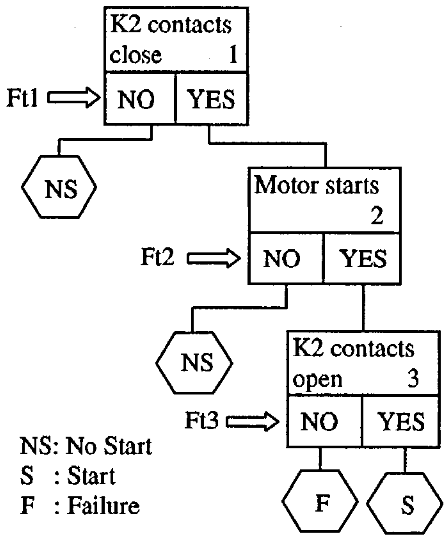


Figure 4.5: Example cause-consequence diagram

operations, which would be the cause of the relay being closed at the start of the next sequence, and later in the sequence it would be unable to open. Andrews and Ridley [6] give an algorithm on how to deal with such events.

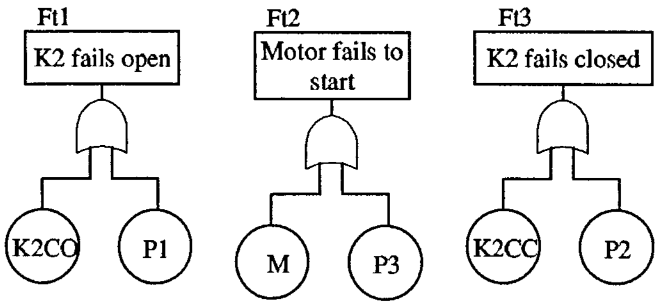


Figure 4.6: Fault trees for the example cause-consequence diagram shown in Figure 4.5

In the example shown in figures 4.5 and 4.6 the relay K2 is required firstly to close (decision box 1) and, later in the sequence, to open (decision box 3). In order to model this type of failure accurately, the cause-consequence diagram requires modification before quantification. A basic event labelling convention in a fault tree structure can be helpful in identifying an inconsistent failure event. If two labels are the same apart from the last character, then they are deemed as inconsistent failure events. This can be seen for the cause-consequence diagram in Figure 4.5, where

Ft1 contains basic event K2CO, the first failure mode, and Ft3 contains the basic event K2CC, the second failure mode.

Following the identification of an inconsistent failure event, the second failure mode is inspected and, depending on whether the second failure mode is an unrevealed or revealed failure event, the cause-consequence diagram is different.

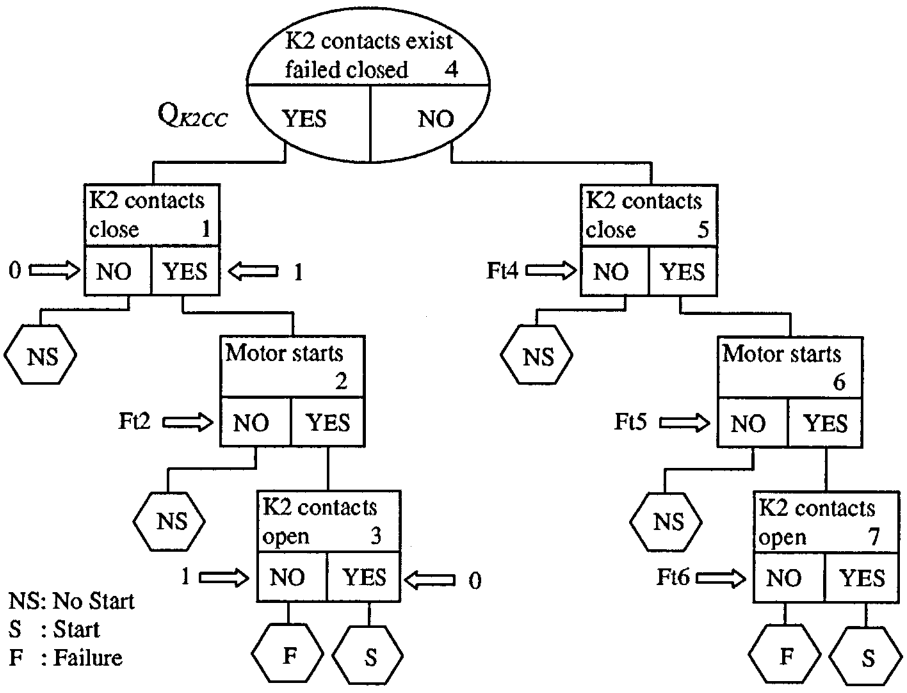


Figure 4.7: Modified cause-consequence diagram for inconsistent failure modes

If the second failure mode is a revealed failure, then it cannot fail between operations and remain undetected. Therefore, the time to failure of the second failure mode is set equal to the time it takes the system to travel from the first failure event to the second failure event. This time will be predicted by the analyst.

If, on the other hand, the second failure mode is unrevealed, then it can occur between operations and be undetected. When this situation occurs, the second failure mode is extracted and placed in an existence decision box preceding the first failure event. The cause-consequence diagram is then duplicated on both outlet branches and, following the YES outlet branch of the existence box, the decision box containing the first failure mode is governed by the failure of the second failure mode. The second failure mode probability is set to 1 in all decision boxes beneath the existence decision box, and the first failure mode is set equal to 0. Following the NO outlet branch of the existence decision box results in the same scenario as if the failure had in fact been a revealed failure.

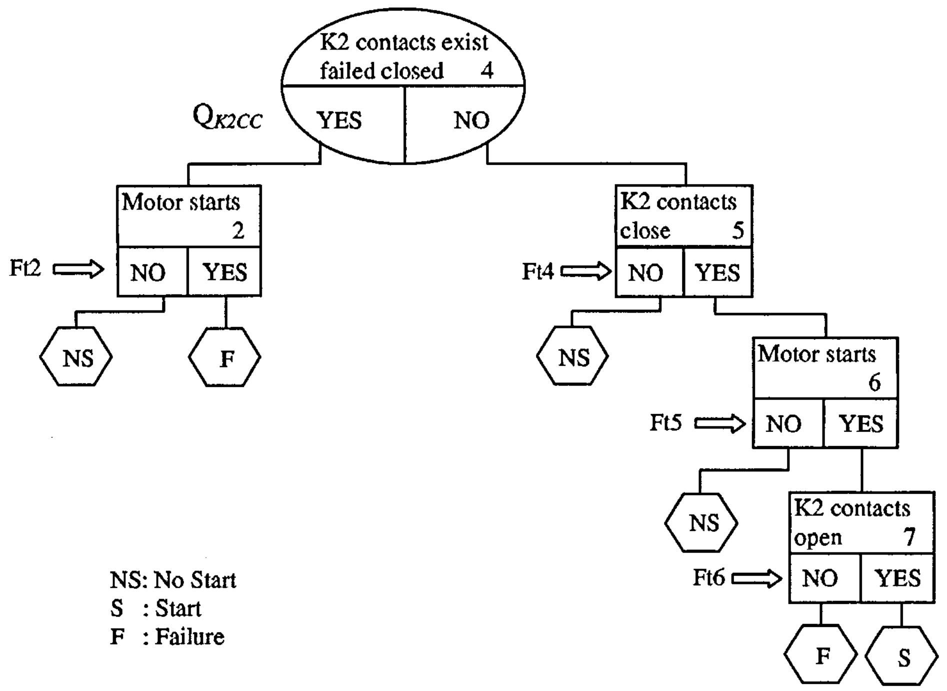


Figure 4.8: Reduced cause-consequence diagram for inconsistent failure modes

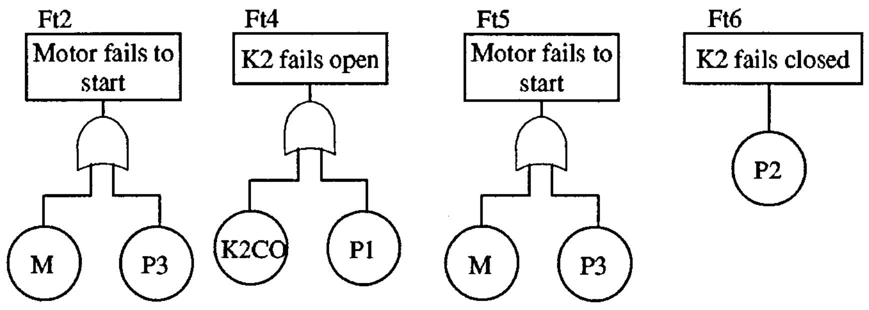


Figure 4.9: Fault trees for the example cause-consequence diagram shown in Figure 4.8

Assuming that K2CC is an unrevealed failure event, the cause-consequence diagram shown in Figure 4.7 would be created and reduced to the form shown in Figure 4.8 with corresponding fault trees shown in Figure 4.9.

Following the inspection of each sequence path in the cause-consequence diagram, and modification due to any identified dependent failure events, the cause-consequence diagram can be quantified by multiplying the probabilities associated with each decision box in each sequence. The probability of any consequence is then obtained via the summation of the probability of any sequence that terminates in that consequence.

4.5 Quantitative analysis

Besides being a tool for analysis of the consequences of critical event the method serves as a basis from which the probability of occurrence of the individual consequence may be evaluated. The cause-consequence analysis can be used as a basis for probability analysis of large complex systems as well as of small, notes Nielsen [5]. Nielsen uses an example of a standby pump system to illustrate how a probability analysis may be carried out. Fault tree analysis was used for the cause part of the diagram. He noted, that special attention should be paid to identify common mode failures. The use of the delay symbol was illustrated - some sequences included integration of the probability distribution function where a failure could occur in a certain time interval. Two different types of events should be considered while evaluating probability of certain consequence: independent and dependent failure events.

4.5.1 Quantitative analysis of a system containing independent failure events

If all events in the system are independent, first of all probabilities are assigned to each outlet branch of the decision box. Then, the probability of every sequence is obtained by multiplying probabilities associated with each decision in that sequence. The final probability of the consequence is obtained by summing probabilities of all sequences ending in that consequence.

To illustrate, the example shown in Section 4.3 can be used. The probability of the event 'No light' will be equal to sum of probabilities of sequences ending in 'NL' (see Figure 4.3). There are two sequences ending in this consequence: one is that the circuit doesn't close and other one is that the circuit closes but there are no current through the filament.

It was said that the circuit will fail to close if the push button fails to close the circuit and probability Q_{PB} was assigned to the NO outlet branch of the decision box 'Circuit closes'. In that case the probability of the first sequence is equal to

Q_{PB}. The probability that circuit will close is equal to $1 - Q_{PB}$ and the probability that there are no current through the filament is Q_{Ft1}. Then the probability of the second sequence is $Q_{Ft1}(1 - Q_{PB})$. The probability of no current through the filament can be obtained using fault tree analysis and it would be equal to $1 - (1 - Q_F)(1 - Q_{BAT})(1 - Q_B)$. Therefore the probability of the consequence 'No light' will be

$$P(\text{No light}) = Q_{PB} + (1 - (1 - Q_F)(1 - Q_{BAT})(1 - Q_B))(1 - Q_{PB})$$

4.5.2 Quantitative analysis of a system containing dependent failure events

Nielsen and Runge [14] gave a procedure to deal with dependent failures, analysing a 2-unit standby system with repair and imperfect switching. A system consists of an operative unit, a switch and a standby unit. The operating unit performs the required system function; when it fails the standby unit is switched into service. It was assumed that switching is done by a human operator.

A consequence diagram shows all relevant events sequentially. In the non-repair situation the diagram is finite because the problem involves only one switchover to the standby. The cause-consequence diagram is shown in Figure 4.10. Element B is a standby element that is switched on if A fails. Therefore, A has to fail before B. If B is unavailable at the time when A fails, the operation ends. The critical event for the cause-consequence diagram A fails in time t_1 doesn't have a cause diagram attached to it to describe its failure causes. This is because the critical event can only be caused by A failing.

To find the probability that the operation stops during time interval 0 to T, $P(T)$, probabilities of sequences 1, 2 and 3 must be determined:

$$P(T) = P_0(T) + P_s(T) + P_f(T)$$

$P_0(T)$, the probability of sequence 1 (that A fails at t_1 AND operator fails to switch over to standby at t_1) is:

$$P_0(T) = K \cdot F_A(T)$$

where K is the probability that operator fails to switch over to standby and F_A is a cumulative failure distribution for the unit A.

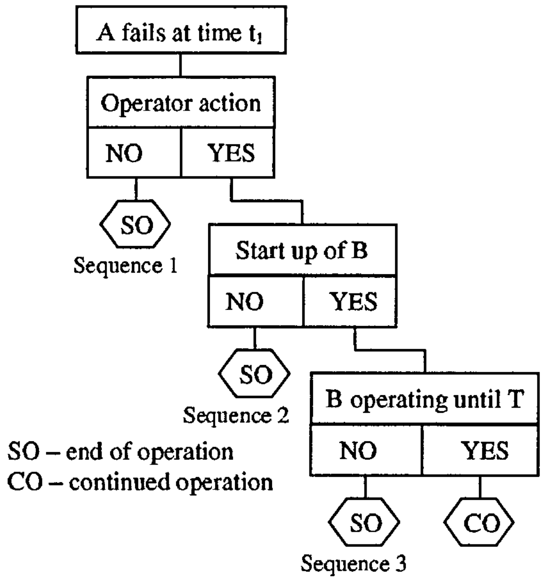


Figure 4.10: Cause-consequence diagram for standby system

The probability of the sequence 2 (A fails at t_1 AND correct operator action at t_1 AND B unavailable at time t_1), $P_s(T)$) is equal to:

$$P_s(T) = \bar{K} \int_0^T f_A(t_1) S_B(t_1) dt_1$$

where $\bar{K}$ is the probability that the operator performs switching action correctly, $f_A(t)$ - the probability density function for time to failure of component A, $S_B(t)$ - the cumulative failure distribution for the standby unit B.

The probability of the sequence 3 (A fails at t_1 AND correct operator action at t_1 AND B available at time t_1 AND B fails during t_1 to T), $P_f(T)$ is described in the same way:

$$P_f(T) = \bar{K} \int_0^T f_A(t_1) \bar{S}_B(t_1) F_B(T - t_1) dt_1$$

where $\bar{S}_B(t)$ is the cumulative distribution that standby unit does not fail during time interval 0 to t_1, F_B - a cumulative failure distribution for the unit B.

The dependent failure was modelled assuming a certain order in which the events occur - component B must be working before it can fail in operation. Cumulative density function $F_B(T-t_1)$ indicates that component B fails in time interval between t_1 and T.

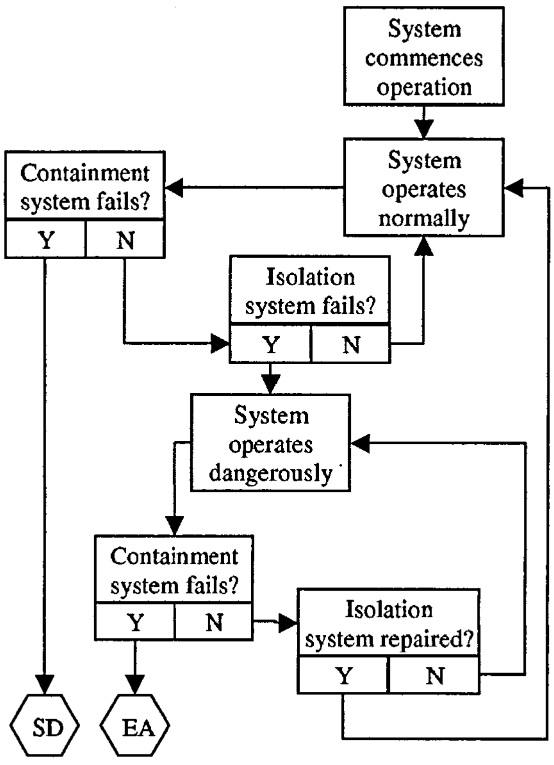
Nielsen and Runge [14] considered the repairable case for the same system as well. Only an approximate result was given as the analysis of the system became too complex for an exact solution. The given model did not account for the importance of the time delays in the system. The model was giving the same failure probability whether the operator doing the switching takes one minute or one hour. The consequences in the last case were said to be usually as severe as for no switching at all.

A different technique is given by Hickling [18]. It was noted that cause-consequence diagrams bear many similarities to flowcharts, which also model various courses of events through a series of decisions. Hickling notes that cause-consequence diagrams could be used to model processes that extend over a period of time using feedback loops: the exit paths from a decision box option are allowed to connect to decision boxes that have already been 'visited'. Each of the feedback loops represent a state of the system. The decision boxes that form that loop each have one option with an exit path that continues around the loop, and another with an exit path that leaves the loop, corresponding to the occurrence of an event which represents a change in the state of the system.

The method was used to model an example system in which the order of failures is important. The cause-consequence diagram for the system is shown in Figure 4.11. The plant consists of two components, a containment system and a leak detection/isolation system which is tested periodically. If the containment system fails first, the isolation system shuts down the plant. Any failure of the isolation system after this is irrelevant. If the isolation system fails first, then until the fault is detected and repair is made, the plant is in a dangerous state in which any failure of the containment system causes total failure.

Hickling states, that because cause-consequence diagrams with feedback loops are no longer representations of a simple Boolean equation, it is not possible to apply the same quantification techniques to them as were described earlier. This is because the input to a decision box is dependent on the output from that box and the output also depends on the input. In this case decision boxes in the loop are associated with failure rates instead of probabilities.

The probabilities of being in the normal and dangerous states at t, $P_N(t)$ and


System commences operation
System operates normally
Containment system fails?
Y
N
Isolation system fails?
Y
N
System operates dangerously
Containment system fails?
Y
N
Isolation system repaired?
Y
N
SD
EA

SD: System shut down
EA: Escape to atmosphere occurs

Figure 4.11: Cause-consequence diagram using feedback loop

$P_D(t)$, can be expressed as:

$$P_N(t) = 1 - P_D(t) - E_S(t) - E_E(t)$$
$$P_D(t) = 1 - P_N(t) - E_S(t) - E_E(t)$$

where $E_S(t)$ is the probability that the system reaches the shut down outcome by t, $E_E(t)$ - the probability that system reaches escape to atmosphere outcome by time t.

The rates at which the system enters the states are expressed as:

$$r_N(t) = P_D(t)\mu_i(t)$$
$$r_D(t) = P_N(t)\lambda_i(t)$$
$$r_S(t) = P_N(t)\lambda_c(t)$$
$$r_E(t) = P_D(t)\lambda_c(t)$$

where $\mu_i(t)$ is the detection/repair rate of the isolation system, $\lambda_i(t)$ - failure rate of the isolation system, $\lambda_c(t)$ - failure rate of the containment system and $r_N(t)$ represents normal operation, $r_D(t)$ - dangerous operation, $r_S(t)$ - shut down, $r_E(t)$ - escape to atmosphere.

The expected number of times that a system enters a certain state by time t is given by:

$$\begin{aligned}
E_N(t) &= \int_0^t P_N(u)\mu_i(u)du \\
E_D(t) &= \int_0^t P_N(u)\lambda_i(u)du \\
E_S(t) &= \int_0^t P_N(u)\lambda_c(u)du \\
E_E(t) &= \int_0^t P_D(u)\lambda_c(u)du
\end{aligned}$$

where $E_N(t)$ is the expected number of times by t that the system enters the normal state of operation, $E_D(t)$ - expected number of times by t that the system enters the dangerous state of operation, $E_S(t)$ - expected number of times by t that shut down occurs, $E_S(t)$ - expected number of times by t that escape to atmosphere occurs.

Hickling notes that this approach can be used with diagrams that have loops with constant exit rates, and where the measures of interest are the relative probabilities of reaching each outcome. In this case the probability of ending in the dangerous state is given by:

$$E_D(\infty) = E_N(\infty)\frac{\lambda_i}{\lambda_c + \lambda_i}$$

which can be solved to give

$$E_D(\infty) = \frac{\lambda_i(\lambda_c + \mu_i)}{\lambda_c(\lambda_c + \lambda_i + \mu_i)}$$

More usefully, the probabilities of the system ending in the state where escape to atmosphere occurs or the system is shut down, are:

$$\begin{aligned}
E_S(\infty) &= \frac{\lambda_c + \mu_i}{\lambda_c + \lambda_i + \mu_i} \\
E_D(\infty) &= \frac{\lambda_i}{\lambda_c + \lambda_i + \mu_i}
\end{aligned}$$

Hickling states that this approach holds much in common with Markov based techniques.

For certain types of systems (particularly those operating sequentially), models can be expressed in a more explicit and transparent way with CCD than with other techniques, states Hickling. With feedback loops, the approach enables the construction of a wider variety of models than would otherwise be possible. The algorithm for the quantification of standard CCD is simpler than that for fault trees, and can be extended directly to CCD with feedback loops.

4.6 Example

For better understanding of the CCA method, the example of the pressure tank system [6] can be used (Figure 4.12). The components individual functions and failure modes are given in Table 4.2. The system contains a start-up, shutdown sequence in addition to its operational phase.

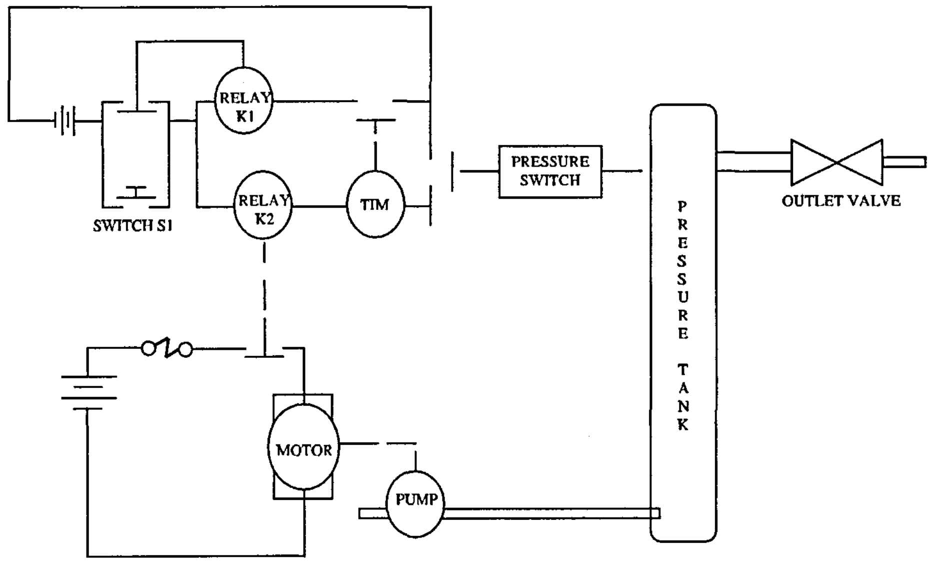


Figure 4.12: Pressure tank system

It is considered that initially the system is de-energized (it is not working). Switch S1 and relay contacts K1 and K2 are all open when the system is in the dormant state, and the timer and pressure switch contacts are closed. Depressing switch S1 provides power to the coil of K1 which results in the closure of the K1 contacts. Relay K1 self-latches when S1 opens when released, and power is also supplied to K2, resulting in K2 contacts closing, which starts the pump motor. It is assumed that the tank takes 30 minutes to fill, and once the pressure threshold is reached the pressure switch contacts open, de-energizing K2, which results in the removal of power from the pump motor. The motor also has a fuse to prevent the power surge, which, if broken, will not allow motor to operate. If the pressure switch fails to open, the timer TIM should time out and the timer contacts open. After a period of time the tank becomes empty and the pressure switch closes, which energizes K2. The pump restarts and the filling process commences again. The tank is filled twice a day and the system is inspected at 6 monthly intervals for dormant failures.

Table 4.2: Component functions and failure modes

Component	Function	Failure modes	Effect on system	Failure type
Switch S1	To apply power to coil of relay K1	S1C: Switch failed closed	Circuit remains energized but can be broken by K2	Unrevealed
		S1O: Switch failed open	No power to energize circuit	Revealed
Relay K1	Electrically self-latched, applying power to relay K2	K1D: Relay fails de-energized	No power to circuit	Revealed
		K1CC: Contact fails closed	Circuit remains energized but can be broken by K2	Unrevealed
		K1CO: Contact fails open	No power to circuit	Revealed
Relay K2	Delivers power to the motor	K2D: Relay fails de-energized	No power to motor	Revealed
		K2CC: Contact fails closed	Continuous power to motor	Revealed
		K2CO:Contact fails open	No power to motor	Revealed
Timer relay (TIM)	Provides emergency shutdown in event of pressure switch failing	TIMCC: Timer contact fails closed	Circuit energized but PRSW can open	Revealed
		TIMCO: Timer contact fails open	No power to motor	Revealed
Pressure switch (PRSW)	De-energizes coil of K2 when tank is full	PSWC: Fails closed	Continuous power to motor	Revealed
		PWSO: Fails open	No power to motor	Revealed
Power supplies 1 and 2	Supplies power to relays and motor	PS1, PS2: No power	No power to motor	Revealed
Motor	Pumps fluid into tank	M: Fails broken	No power to motor	Revealed

Three steps are considered by Andrews and Ridley [6]:

Step1 Component failure event ordering. The ordering of the components for the construction of the cause-consequence diagram is selected by considering the temporal patterns of the system. For the pressure tank system, switch S1 is depressed, followed by its opening. Relay K1 energizes and powers K2 which powers the pump. Following 30 minutes of operation, the pressure switch should open. In the event that the pressure switch fails to open, the timer should time out and the timer contacts open. Given that the pressure switch opens, K2 contacts should de-energize, removing power from the pump. Where the timer is required to break the circuit containing K1, K1 contacts should de-energize, removing power from K2, which results in the removal of the power supply to the pump. The ordering was therefore chosen to be

S1, K1, K2, pressure switch, timer relay, K1, K2

It can be seen that the components K1 and K2 both occur twice in the ordering sequence. This is the result of the system containing two different phases, and hence some components perform different actions in each different phase. The components K1 and K2 are both required to be closed in the startup sequence and open in the shutdown sequence.

Steps 2 and 3 Cause-consequence diagram construction and reduction. The cause-consequence diagram was constructed by considering the effect of each component in the chosen order on the system performance. In order to highlight relevant features, only one filling sequence is investigated, the cause-consequence diagram of which is given in Figure 4.13. The corresponding fault trees are illustrated in Figure 4.14.

4.6.1 System quantification

Prior to multiplying the probabilities associated with each decision box in each sequence, the cause-consequence diagram was checked for any dependent failure events [6]. The following dependent failure events were identified.

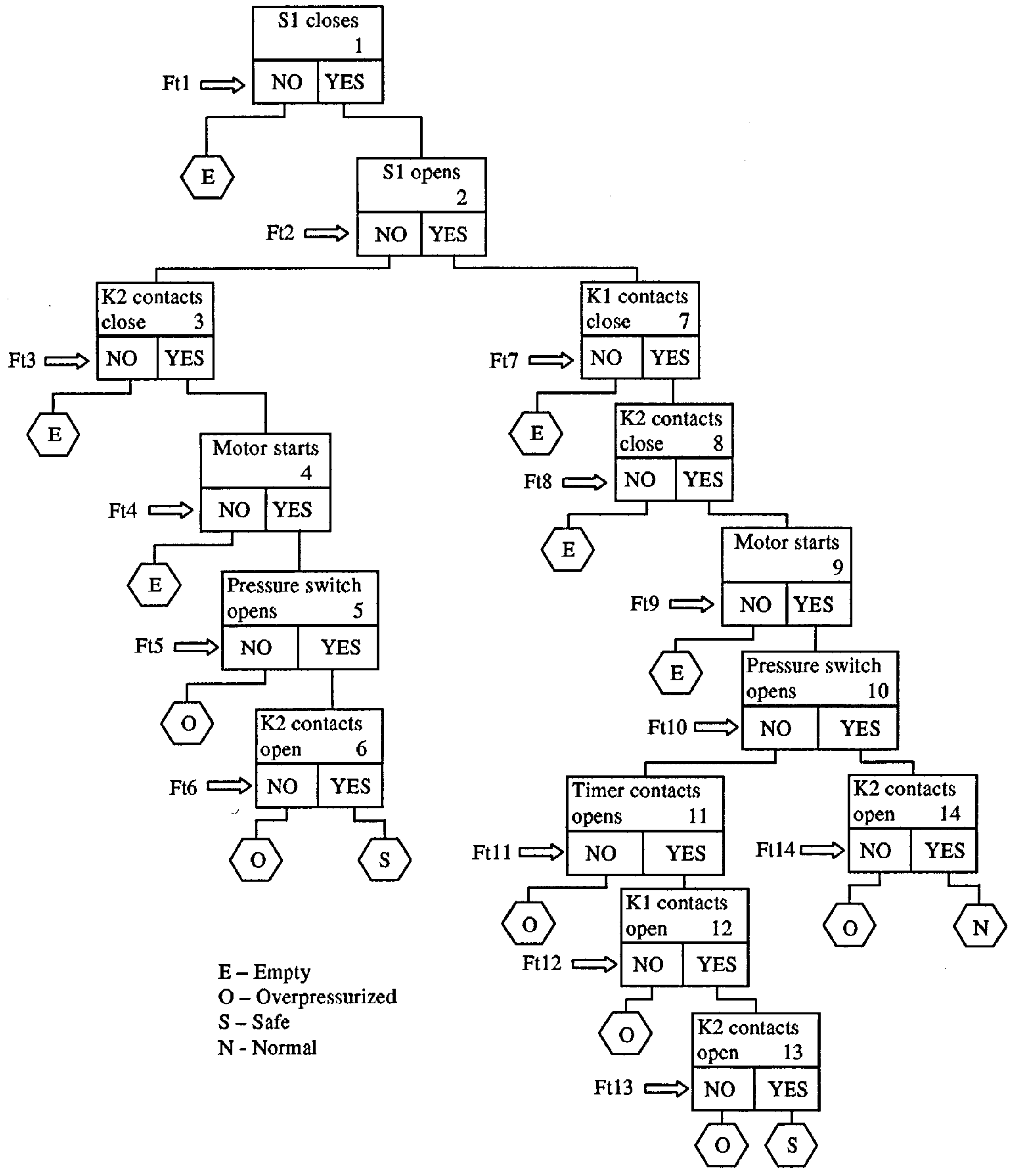


Figure 4.13: Cause-consequence diagram for the pressure tank system

1. Inconsistent failure event present in Ft1 and Ft2 as the switch is required to close, represented by decision box 1, and then open, represented by decision box 2. The second failure event, S1FC, is an unrevealed failure event (Table 4.2) and is therefore extracted and placed in an existence decision box preceding decision box 1. The cause-consequence diagram is modified using

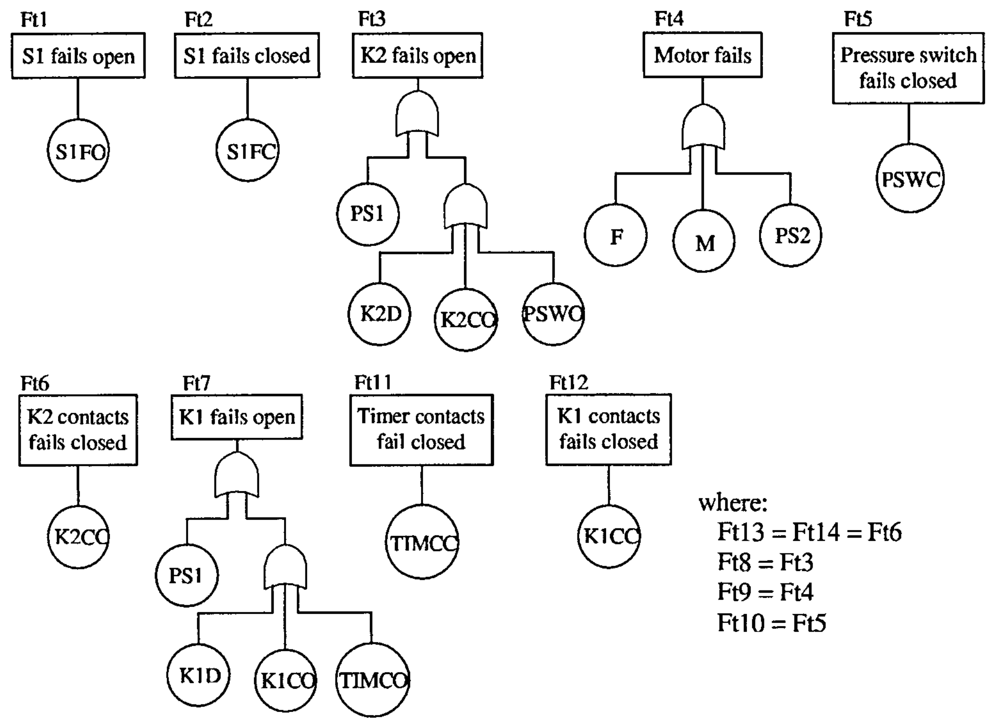


Figure 4.14: Fault trees for the pressure tank cause-consequence diagram

the procedure detailed in the Section 4.4.3.2.

2. Inconsistent failure event present in Ft3 and Ft5 as the pressure switch is required to be closed and then open. The second failure event, PSWC, is a revealed failure event (Table 4.2) and the time to failure of PSWC is set equal to 30 minutes (the filling time).

3. Inconsistent failure event present in Ft3 and Ft6 as K2 contacts are required to close and, following the tank being full, open. The second failure event, K2CC, is a revealed failure event (Table 4.2) and the time to failure of K2CC is set equal to 30 minutes (the filling time).

4. Common failure event present in Ft7 and Ft8, PS1 is extracted and placed in a new decision box preceding decision box 7. The cause-consequence diagram is modified following the procedure detailed in Section 4.4.3.1.

5. Inconsistent failure event present in Ft7 and Ft12 as K1 contacts are required to close and then open. The second failure event, K1CC, is an unrevealed failure event (Table 4.2) and is therefore extracted and placed in an existence

decision box. The cause consequence diagram is modified using the procedure detailed in Section 4.4.3.2.

6. Inconsistent failure event present in Ft7 and Ft11 as the timer contacts are closed and may be required to open later in the sequence. The second failure event, TIMCC, is an unrevealed failure event (Table 4.2) and is therefore extracted and placed in an existence decision box. The cause-consequence diagram is modified using the procedure detailed in Section 4.4.3.2.

Following the appropriate modification owing to the dependent failure events identified, the final cause-consequence diagram was developed and is shown in Figures 4.16 and 4.17, with corresponding fault trees given in Figure 4.15.

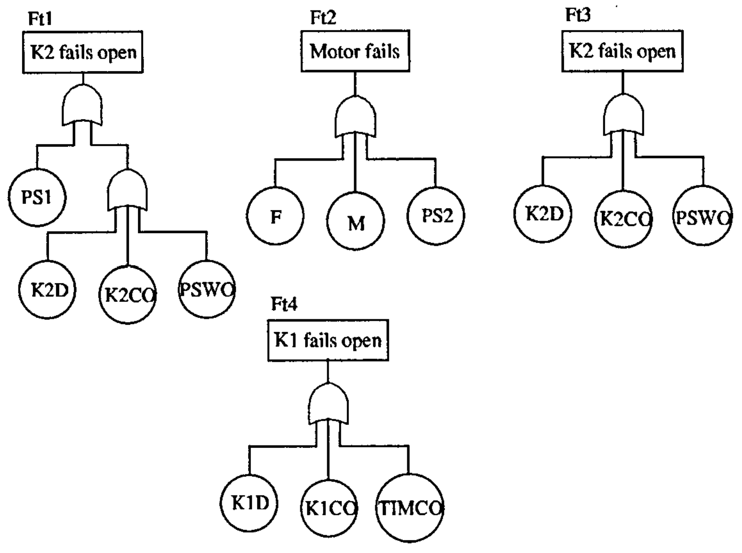


Figure 4.15: Fault tree structures for Figures 4.16 and 4.17

The system functions twice daily and therefore the time between operations is $12h$. The probability of failure for revealed failures between operations was hence obtained using equation (4.1) with $t = 12h$. For unrevealed failures the probability of the failure was obtained using θ and τ, given in Table 4.3, and equation (4.2) (λ is conditional failure rate):

$$Q = 1 - e^{\lambda t} \tag{4.1}$$

$$Q_{AV} = \lambda(\frac{\theta}{2} + \tau) \tag{4.2}$$

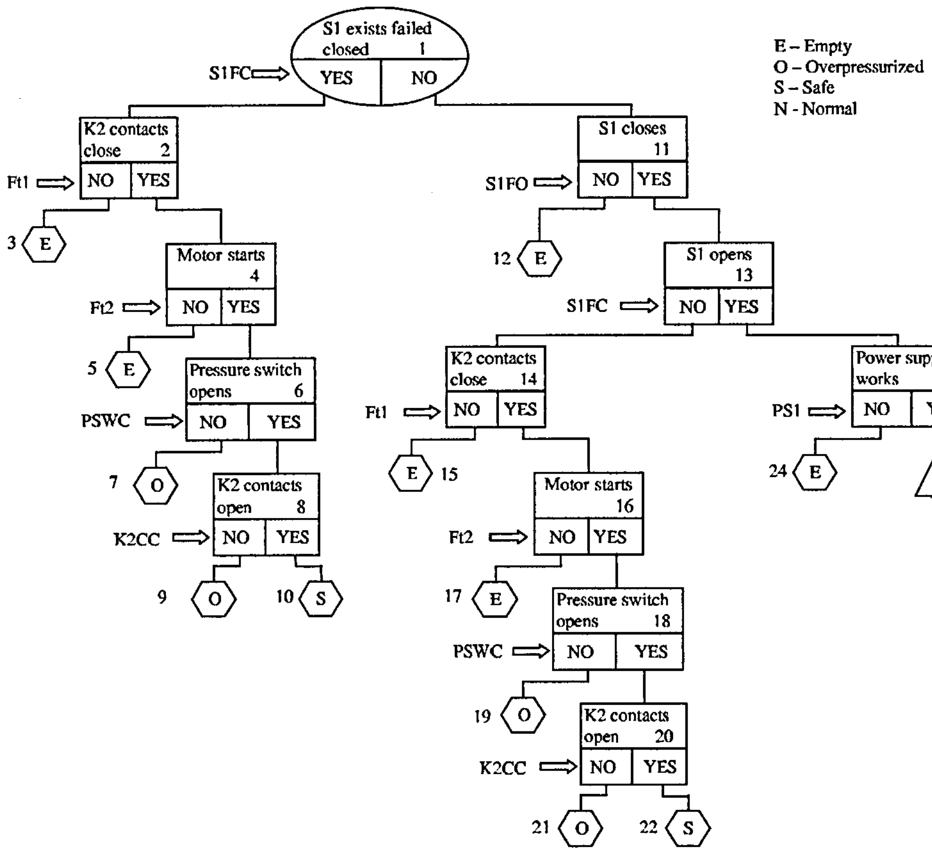


Figure 4.16: First page of the final cause-consequence diagram for the pressure tank system

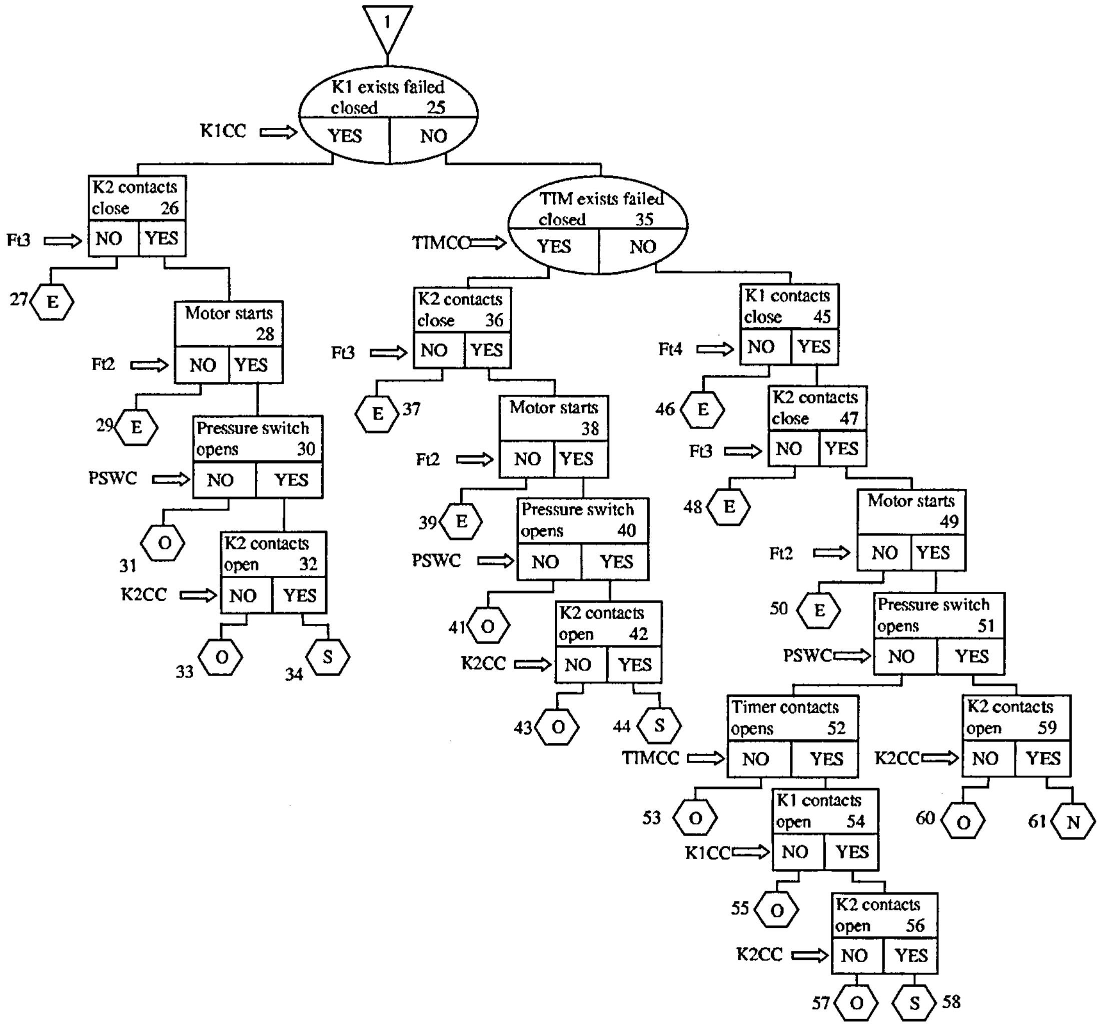


Figure 4.17: Second page of the final cause-consequence diagram for the pressure tank system

The probability of each fault tree was calculated using the inclusion-exclusion method, and the probability of overpressure was obtained by summing the probabilities of any sequence that terminated in the consequence 'O'. There existed 12 such paths. In addition to obtaining the probability of overpressure, the probability of the tank being empty, a safe operation and normal operation was also calculated.

Table 4.3: Component functions and failure modes

Component	Failure rate	Inspection interval, θ	Mean time to repair, τ
Switch S1	S1FC: 1×10^{-6}	4368.0	36.0
	S1FO: 8.698×10^{-4}	NA	NA
Relay K1	K1D: 0.23×10^{-6}	NA	NA
	K1CC: 0.23×10^{-6}	4368.0	36.0
	K1C0: 0.23×10^{-6}	NA	NA
Relay K2	K2D: 0.23×10^{-6}	NA	NA
	K2CC: 0.23×10^{-6}	NA	NA
	K2C0: 0.23×10^{-6}	NA	NA
Timer relay	TIMCC: 1×10^{-4}	4368.0	36.0
	TIMCO: 1×10^{-4}	NA	NA
Pressure switch	PSWC: 1×10^{-4}	NA	NA
	PSWO: 1×10^{-4}	NA	NA
Fuse	F: 1×10^{-5}	NA	NA
Power supplies 1 and 2	PS1: 1×10^{-6}	NA	NA
	PS2: 1×10^{-6}	NA	NA
Motor	M: 1×10^{-6}	NA	NA

4.7 Applications of Cause-Consequence Diagram Method

Several authors have applied the method to various systems. In 1976, Burdick and Fussell [16] made a first step in adapting CCA to standardised use in the US nuclear power industry. They also stated, that the cause-consequence analysis should be combined with other new methods of analysis, such as phased mission analysis. The application of CCD for a 2-unit standby system with repair and imperfect switching was carried out by Nielsen and Runge in 1974 [14]. They investigated both repairable and non-repairable cases. For the repairable case, analysis of the system was more complex and only an approximate probability of failure was given. In 1975 Nielsen, Platz and Runge [19] used CCD method to analyse a redundant protection system. The protection system analysed was a core spray system in a nuclear boiling water reactor and it was used to prevent the fuel core from overheating given a loss of primary coolant. Reliability of a proposed instrument air system for a complex system of fertilizer plants was studied by Nielsen, Platz and Kongso [24] (1977). Using the CCD method they pointed out inadequate system designs and identified useful design changes to improve reliability of the system. The CCD method was also applied to design interlocks (arrangements of switching components designed to prevent operating signals being sent to plant components in dangerous circumstances) by Tailor [25] in 1976. More recently, Andrews and Ridley [6, 23] applied the CCD method to sequential systems as well as to static systems.

4.8 Summary

Following Nielsen [5], the cause-consequence diagram method should be regarded as a tool by which problems are defined and presented and it could also serve as a basis from which the probability of occurrence of the individual consequences may be evaluated. Since the early work the method has been extended and adopted to model various industrial systems.

One of the advantages of the cause-consequence diagram is that it identifies the complete set of system responses to any given initiating event. This can be achieved using event trees as well, but the cause-consequence diagram is able to model more complex events, i.e. dependent events. Unlike fault trees, the cause-consequence diagram method retains failure logic for the system and it is possible to develop the diagram from system logic.

5. COMPONENT ORDERING STRATEGIES

5.1 Introduction

Before a BDD is constructed, basic events in the fault tree need to be ordered. Depending on the chosen ordering, the size of BDD and complexity of the calculations required for its construction can change dramatically. Previous research outlined different ordering strategies for basic events in the fault tree.

In this chapter two main groups of ordering schemes are discussed - structural and weighted ordering techniques. Structural ordering schemes involve ordering the variables via a structured traversal of the fault tree and they have a tendency to keep close in the ordering scheme those variables that appear close together in the fault tree. The most common ordering technique is the top-down scheme, which is described first. Other structural ordering techniques include modified top-down, depth-first, modified depth-first, modified priority depth-first and depth-first with number of leaves.

Weighted ordering techniques work slightly differently by allocating weights to the variables and then determining their position in the ordering. These schemes can be divided into topological schemes, which assign weights according to the position of the variable in the fault tree, and the ones based on importance measures (event criticality was used an example). Non-dynamic top-down weights, dynamic top-down weights and bottom-up weights represent topological weighted ordering schemes.

5.2 Structural Ordering Schemes

5.2.1 Top-Down Ordering

The top-down scheme orders variables as they appear in a fault tree following top-down, left-right traversal of the fault tree structure. Therefore basic events

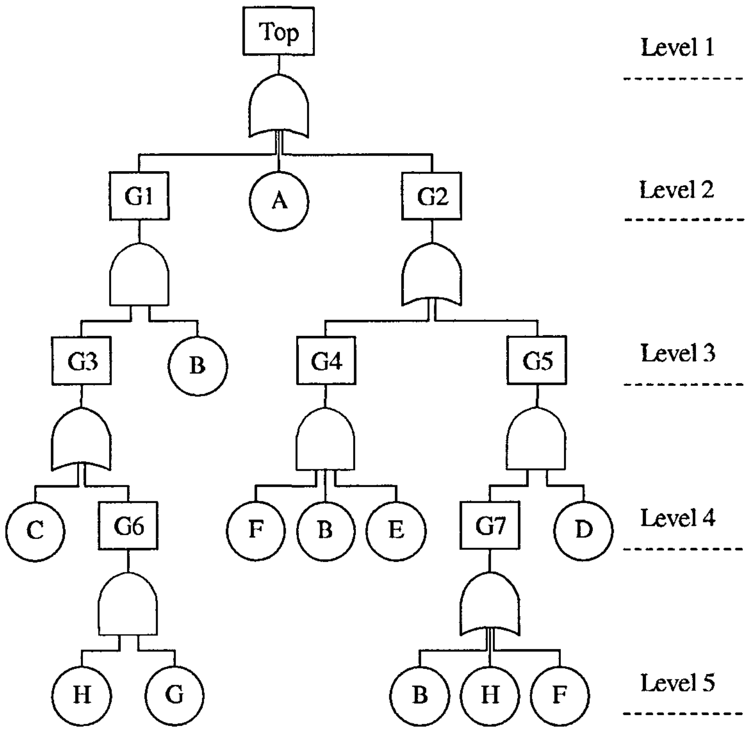


Figure 5.1: Example fault tree used for ordering

appearing on the higher levels of fault tree will be placed earlier in the ordering than those appearing lower down the fault tree.

To illustrate how this works, the scheme can be applied to the fault tree shown in Figure 5.1. Each level is considered in turn, from the top of fault tree going downwards, and the basic events are ordered from left to right on each level. Each event is placed in the ordering the first time it is encountered in the fault tree and subsequent occurrences of the particular basic event are ignored.

Following the top-down approach the ordering of the basic events for the fault tree shown in Figure 5.1 is:

$$A < B < C < F < E < D < H < G$$

This scheme is highly dependent on the way the fault tree is structured. For example, if gates G1 and G2 where swapped around, or the order of basic events as inputs to the gates was changed, then the order in which basic events are placed would change as well, although the logic function of the fault tree remained the same. These changes could affect the size of the resulting binary decision diagram or cause-consequence diagram.

5.2.2 Modified Top-Down Ordering

Using the modified top-down ordering scheme the fault tree is scanned in the same way as in top-down approach - the basic events appearing on higher levels are placed in the ordering before the basic events appearing on the lower levels of the fault tree. The basic events appearing on the same level of the fault tree are ordered not in just left-right order, but also according to their total number of occurrences throughout fault tree: the basic events appearing more often will be placed in the ordering first. If there are two or more basic events that appear the same number of times in the fault tree, they are ordered from left to right as they occur on that level. Each variable is placed in the ordering scheme as it is first encountered on the fault tree and any subsequent appearances are ignored.

For the example fault tree in Figure 5.1 the modified top-down ordering is:

$$A < B < F < C < E < D < H < G$$

5.2.3 Depth-First Ordering

The depth-first ordering scheme considers the fault tree to be made up of many smaller subtrees, and each subtree is ordered in top-down, left-right manner. Starting from the top of the fault tree, basic event inputs are placed in the ordering as they appear from left to right, before considering any gate inputs. The gate inputs are considered from left to right and each of them is then considered as the top event and ordered the same way, such that the lower levels of the most-left subtree are placed in the ordering before higher levels of the other subtrees.

For the example fault tree in Figure 5.1 the depth-first ordering is:

$$A < B < C < H < G < F < E < D$$

5.2.4 Modified Depth-First Ordering

The modified depth first ordering scheme considers the gate inputs to any gate in a left-right manner, the same way as the depth-first ordering scheme, such that the subtree of a left-most gate is completely explored before considering the remaining gate inputs and any basic event inputs to a gate are considered before the gate inputs. The difference is that basic events on the same level of a subtree are ordered according to the number of their appearances in the fault tree. The events with the higher number of appearances are placed in the ordering first, but if there are two or more events that appear in the fault tree the same number of times, then they are ordered as they appear from left to right in the gate.

For the example fault tree in Figure 5.1 the depth-first ordering is:

$$A < B < C < H < G < F < E < D$$

5.2.5 Modified Priority Depth-First Ordering

This ordering scheme is an extension of the modified depth-first ordering, where rather than simply considering the gate inputs from left to right, any gates, which themselves have only basic events as inputs, are considered first. Basic events are ordered as in the modified depth-first ordering scheme, such that the most repeated events are given priority and, if there is a tie, then they are ordered from left to right as they appear in the list of inputs. Basic events continue to be considered before any gate inputs.

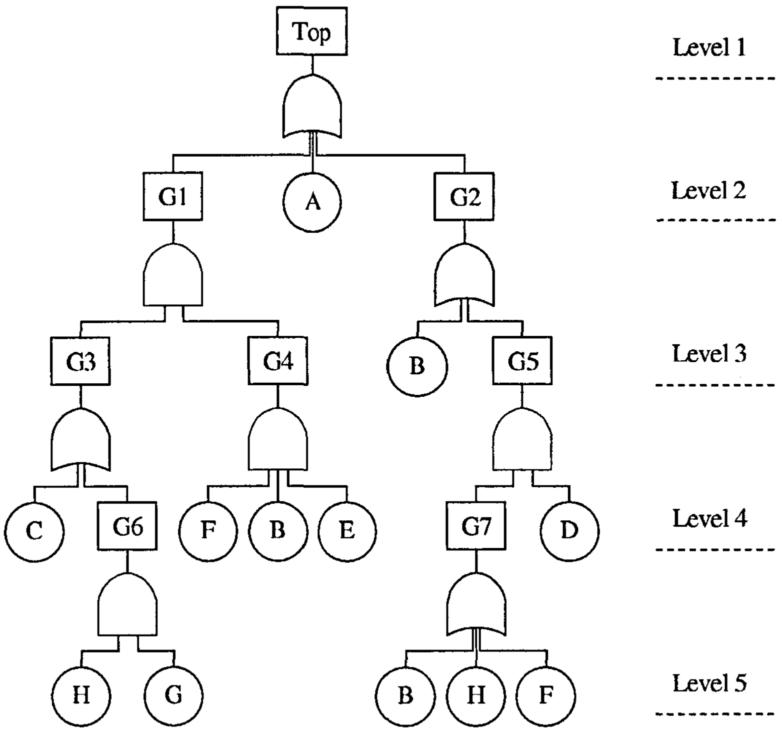


Figure 5.2: Example fault tree used for ordering

To illustrate this ordering scheme consider the example fault tree shown in Figure 5.2. The top event has three inputs - gates 'G1' and 'G2' and basic event

'A'. Basic event 'A' is placed in the ordering first. Gate 'G1' has two inputs - gates 'G3' and 'G4'. Inputs to gate 'G2' are basic event 'B' and gate 'F5'. These gates are investigated from left to right. Inputs to gate 'G3' are basic event 'C' and gate 'G6' and to gate 'G4' - basic events 'F', 'B' and 'E'. As gate 'G4' has only basic events as its inputs, it is considered first. Basic event 'F' occurs twice in the fault tree, 'B' - three times, and basic event 'E' appears only once in the fault tree. Therefore, the next basic event placed in the ordering is 'B' and it is followed by basic events 'F' and 'E'. This gives partial ordering $A < B < F < E$. Gate 'G3' has one basic event and one gate as its inputs. Basic event 'C' is placed in the ordering first. Following this, basic events from gate 'G6' are considered. Basic event 'H' is repeated twice in the fault tree, and basic event 'G' appears once. Therefore, basic event 'H' is placed in the ordering before basic event 'G'. This gives the partial ordering $A < B < F < E < C < H < G$. Next gate to consider is 'G2'. It has basic event 'B' as an input, but it has already been placed in the ordering. Inputs to gate 'G5' are gate 'G7' and basic event 'D'. Basic event 'D' is placed in the ordering first and then inputs to gate 'G7' are considered. In this case, all three basic events of gate 'G7' have already been placed in the ordering. This gives the final ordering:

$$A < B < F < E < C < H < G < D$$

5.2.6 Depth-first, with Number of Leaves

This is another ordering scheme that is an extension to the modified depth-first ordering. It uses a different method than the modified priority depth-first ordering to choose the order in which gate inputs are explored. In this case gates are considered according to the number of 'leaves' beneath the gate itself. The number of leaves of a gate is the total number of basic events occurring at any level beneath that gate.

The gate inputs with the least number of leaves that have not been ordered are considered first. In the case of a tie, the gate with fewest ordered leaves is chosen. If an order still can't be established, then they are placed in the ordering as they appear from left to right in the fault tree. The basic events are ordered the same way as in modified depth-first ordering, so the most repeated events are chosen first. In the case of a tie, they are ordered as they appear from left to right. Basic events are placed in the ordering before any gates.

For the example fault tree in Figure 5.1 the number of leaves for each gate is shown in Table 5.1.

Gate	G1	G2	G3	G4	G5	G6	G7
Number of leaves	4	7	3	3	4	2	3

Table 5.1: Number of leaves of each gate in Figure 5.1

To illustrate how this ordering works, start with top event. The top event has three inputs - basic event 'A' and gates G1 and G2. Basic event 'A' is placed in the ordering first as it has fewer leaves than either G1 or G2, then gate G1 is considered first as it has fewer leaves (4) than G2 (7). This gives partial ordering $A < B < C < H < G$. Basic events within the gate are ordered simply as they appear from left to right. After gate G1, gate G2 is considered next and it has two gate inputs - G4 and G5. Gates G4 and G5 both have two unordered leaves, but because gate G4 has only one ordered leave ('B') while gate G5 has two ordered leaves ('B' and 'H'), it is processed first. The partial ordering at this point is $A < B < C < H < G < F < E$. Gate G5 has input 'D' which is placed in the ordering next. As all basic events are already placed in the ordering, gate G7 has nothing further to add to the list. The final ordering is:

$$A < B < C < H < G < F < E < D$$

5.3 Weighted Ordering Schemes

5.3.1 Non-Dynamic Top-Down Weights

Non-dynamic top-down weights ordering scheme places basic events in the order of decreasing weight. Weights are calculated for each event according to the following steps:

- A weight of 1.0 is assigned to the top event and is propagated through the fault tree towards the basic events.
- At each gate, the weight is equally distributed between its inputs.
- Each basic event will then be assigned a weight. Repeated events have their corresponding weights added together.

- The highest order is given to the basic event with the largest weight.

For events with equal weights their average level of appearance in the fault tree is calculated. It is obtained by summing the levels on which events occur and dividing this by the number of occurrences. The basic event that appears, on average, highest in the tree is placed earlier in the ordering. If basic events still tie for position then the most repeated event is chosen and if a tie still exists then they are simply ordered as they appear in the modified top-down ordering.

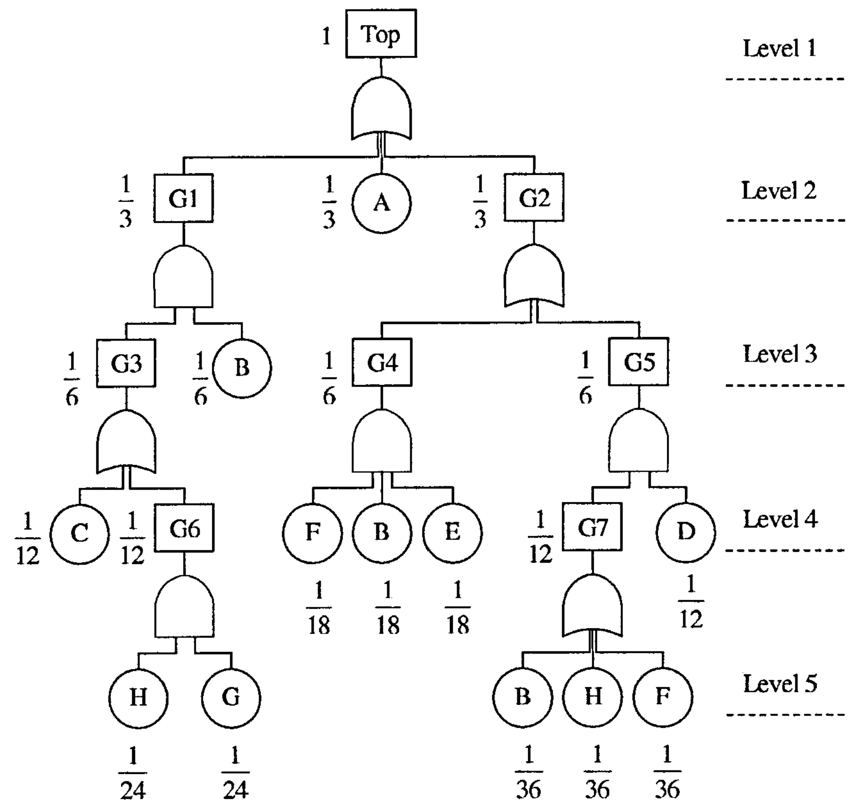


Figure 5.3: Fault tree from Figure 5.1 after assigning weights

To illustrate how this ordering scheme works, it is applied to a fault tree shown in Figure 5.1. The fault tree after weights have been assigned is shown in Figure 5.3.

Next, the weights of each basic event are calculated:

$$\begin{aligned}
A &= \frac{1}{3}\\
B &= \frac{1}{6}+\frac{1}{18}+\frac{1}{36}=\frac{1}{4}\\
C &= \frac{1}{12}\\
D &= \frac{1}{12}\\
E &= \frac{1}{18}\\
F &= \frac{1}{18}+\frac{1}{36}=\frac{1}{12}\\
G &= \frac{1}{24}\\
H &= \frac{1}{24}+\frac{1}{36}=\frac{5}{72}
\end{aligned}$$

After the weights are calculated, the first basic event to be placed in ordering is 'A' and then it is 'B'. Basic events 'C', 'D'and 'F' have the same weight. Events 'C' and 'D' appear on the same level (Level 4), and 'F' appears on levels 4 and 5, therefore basic events 'C' and 'D' are placed in the ordering first. As they have the same weight and appear on the same level, they are placed as they appear in the fault tree from left to right, so basic event 'C' is placed first. The final ordering is:

$$A < B < C < D < F < H < E < G$$

5.3.2 Dynamic Top-Down Weighted Ordering

Dynamic top-down weighted ordering calculates weights of the basic events the same way as the non-dynamic version, but only the event with the highest weight is placed in the ordering. Once an event has been placed in the ordering, it is then removed from the fault tree by deleting all its occurrences. Using the modified fault tree weights are reassigned. This allows another basic event to be placed in the ordering and the process continues until all events have been ordered.

From the example in Figure 5.3, the first event to place in the ordering is event 'A'. Then this basic event is removed from the fault tree and the resulting fault tree is shown in Figure 5.4.

Now the basic event that has the largest weight is 'B'. Therefore it is placed in the ordering and removed from fault tree. The procedure is repeated until all basic

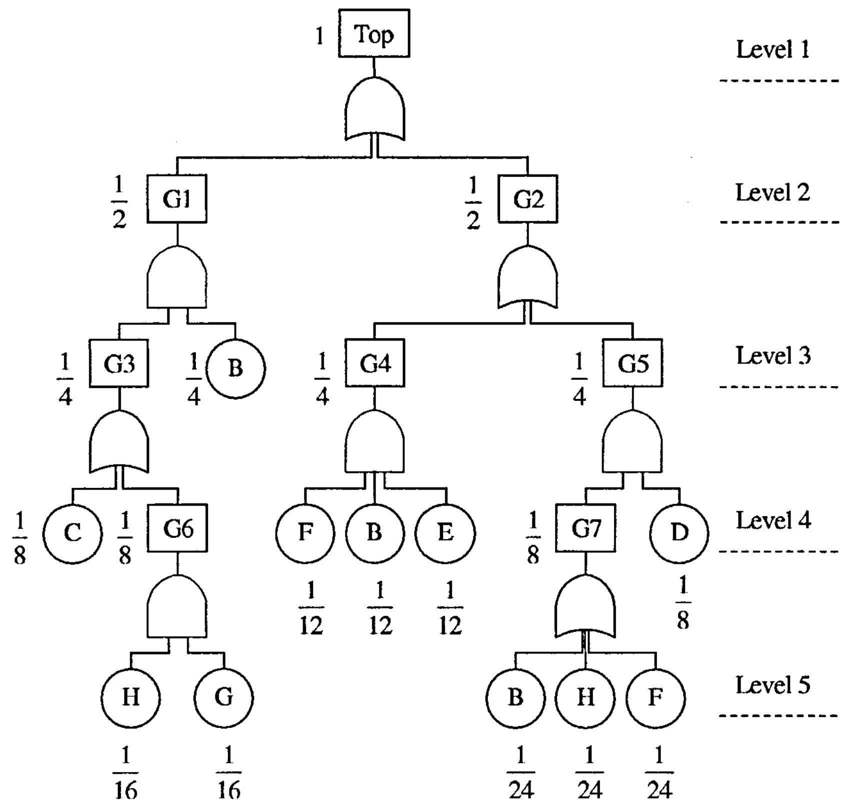


Figure 5.4: Modified fault tree from Figure 5.3

events are placed in the ordering. The final ordering is:

$$A < B < C < H < G < F < E < D$$

5.3.3 Bottom-Up Weights

Bottom-up weighted ordering starts from the bottom of the tree, rather than the top and in effect calculates weights for the gates, which are then used to determine the ordering in which they are considered within a depth-first exploration. The main features are:

- a weight of 1/2 is assigned to each basic event and propagated towards the top event.

- at each gate, the weights of the inputs are combined as probabilities according to:

$$\begin{aligned}\text{'AND' gate: } P\,(\text{gate}) &= \prod_{i=1}^{n} q_{w_i} \\ \text{'OR' gates: } P\,(\text{gate}) &= 1 - \prod_{i=1}^{n} (1 - q_{w_i})\end{aligned}$$

where n is the number of inputs to the gate and q_{w_i} is the weight of basic event i.

- Once each of the inputs to the top event has been assigned weights, the tree is explored in a depth-first manner, considering branches with the largest weight first.

Once the weight values of the the gates have been established, the method proceeds as in the modified depth-first method, except that the gates are explored according to which has the highest weight rather than simply from left to right. However, if gates do have the same weight then they are considered according to the percentage of repeated events below that gate. This is calculated by adding up the number of repeated events below the gate and dividing by the total number of events below that gate. The gate with the highest number of repeated events is considered first, but if there is a tie, then they are considered from left to right as they appear in the input list. The basic events of each gate are ordered before the gate inputs are explored and are chosen according to the highest number of occurrences in the fault tree. If events have the same number of occurrences then they are simply chosen from left to right as they appear in the input list.

To illustrate how this scheme works weights are assigned to basic events of the fault tree in Figure 5.1 and the new fault tree is shown in Figure 5.5. As basic events are ordered before gate inputs, basic event ‘A’ is placed in the ordering first. Gate G2 is considered next as it has larger weight than gate G1. There are two inputs to gate G2 - G4 and G5. Gate G5 is investigated first as it has larger weight. Next basic event placed in the ordering is ‘D’ and then basic events ‘B’, ‘H’ and ‘F’. The final ordering is:

$$A < D < B < H < F < E < C < G$$

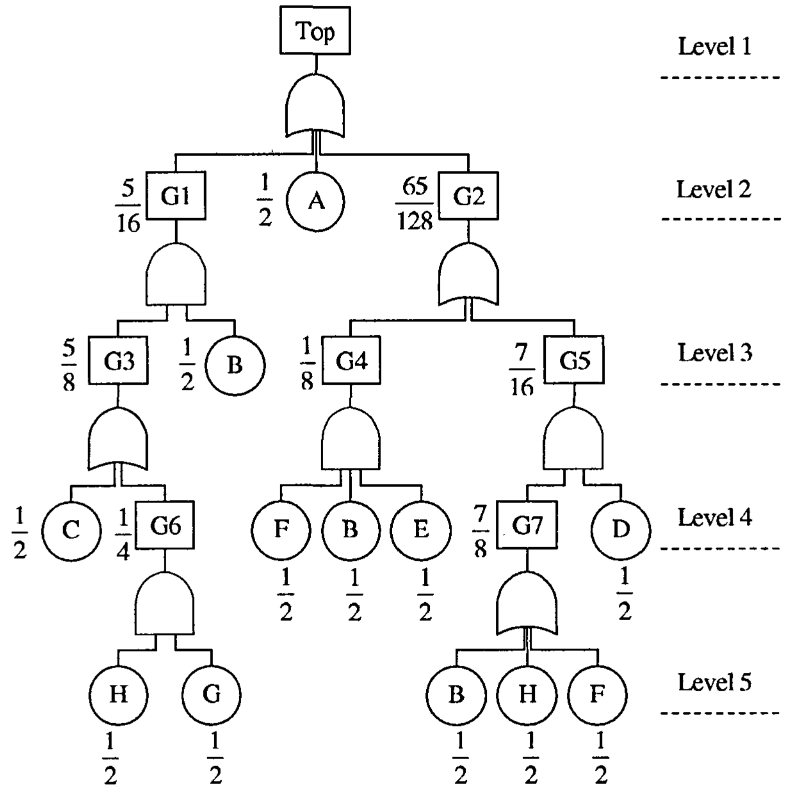


Figure 5.5: Fault tree from Figure 5.1 after assigning weights

5.3.4 Event Criticality

This final ordering scheme to be considered is an extension of the one that applies the principle of Birnbaum's structural importance measure directly to the tree. The contribution of each basic event to the top event is calculated according to:

$$I_i = Q\left(1_i, \underline{1/2}\right) - Q\left(0_i, \underline{1/2}\right)$$

The selected basic event therefore assumes the failure probabilities of one and zero on two consecutive computations of the top event probability, with the remaining components given failure probabilities of 1/2. The result of the second run (with failure probability zero) is subtracted from the first run (with failure probability one) to give the contribution of that basic event to occurrence of the top event.

The basic events are ordered such that those with a greater contribution to the occurrence of the top event are ordered before these with smaller contributions. If two events have the same calculated contribution, then the event with the highest average level of occurrence is selected first. If the events are still tied then the most repeated event is selected and if the events are still indistinguishable, then they are simply ordered as they appear in the modified top-down ordering.

For then example fault tree in Figure 5.1 the ordering following this method would be:

$$A < D < B < C < F < H < E < G$$

5.4 Summary

The ordering schemes described above will be used to investigate the influence of the order of the failure events to the size of the cause-consequence diagram. Previous research showed that the order of basic events can have a big influence to the size of binary decision diagrams.

It has to be noticed, that variable orderings, produced by each of the schemes, are very sensitive to the way the fault tree is written. The structure of fault tree can vary significantly without any difference in the structure function. Also, fault trees often are not written in minimal form, which would affect both the ordering of the basic events, and the size of the resulting binary decision diagram or cause-consequence diagram.

6. REVIEW OF PHASED MISSION ANALYSIS METHODS

6.1 Introduction

One of the most important problems in system unreliability is the phased mission problem [27]. Most reliability analysis techniques and tools assume that a system is used for a mission consisting of a single phase [28, 29]. However, multiple phases are natural in many missions. With increasing complexity and automation associated with the systems encountered in the nuclear, aerospace, chemical, electronic, and other industries, phased mission analysis is being recognized as the appropriate reliability analysis method for a large number of problems [1].

Many systems perform a mission which can be divided into consecutive time periods - phases. The phase duration may be fixed or random. In each phase, the system needs to accomplish a specific task. The system configuration (the logic model), the phase duration, and the failure rates of the components often vary from phase to phase [28].

The following is a description of a phased mission [1]:

> *A phased mission is a task to be performed by a system during the execution of which the system is altered such that the logic model changes at specified times. Thus, during a phased mission, time periods (phases) occur in which either the system configuration, system failure characteristics, or both, are distinct from those of any immediately succeeding phase.*

A classic example of a phased mission system is an aircraft flight which involves take-off, ascent, level flight, descent and landing phases.

6.2 Analysis of phased mission systems

Different types of phased mission systems occur and each of them has its own phased mission analysis problems [1]. The components of the system may fail independently of each other or have interdependent failure properties. The components may be repairable, with specified repair times, or they may be nonrepairable. Often a system undergoing a phased mission will contain both repairable and nonrepairable components. In a mission such as an intercontinental ballistic missile, all components will be nonrepairable. During a manned space flight, however, it may be possible for an astronaut to replace or repair a malfunctioning item [27].

The most important phased mission analysis problem is to calculate exact or, obtain bounds for, mission unreliability, where mission unreliability is defined as the probability that the system fails to function successfully in at least one phase [1, 27]. Estimating the mission reliability by the product of the phases usually results in an appreciable overprediction in system reliability, since basic events are shared among the logic models of the various phases.

An example of this is given by Esary and Ziehms [2]. A system with two independent components, C_1 and C_2, is designed for a two-phased mission. In order for the system to perform the required tasks at least one component has to function through phase 1 and both components have to function through phase 2. The block diagrams for this are shown in Figure 6.1.

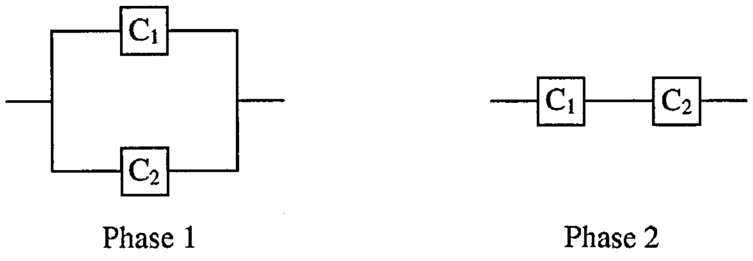


Figure 6.1: Block diagram for two-phased mission

The probabilities of the components are as follows: P_{11} - probability that component C_1 functions through phase 1; P_{21} - probability that component C_2 functions through phase 1; P_{12} - probability that component C_1 functions through phase 2, given that it has functioned through the phase 1; P_{22} - probability that component C_2 functions through phase 2, given that it has functioned through phase 1.

The system reliability for the phase 1 R_1 is given by

$$R_1 = P_{11} + P_{21} - P_{11}P_{21}$$

and system reliability for phase 2, R_2, given that both components have functioned through phase 1, is

$$R_2 = P_{12}P_{22}.$$

Multiplying these together would lead to the mission reliability

$$R = R_1R_2 = (P_{11} + P_{21} - P_{11}P_{21})P_{12}P_{22}.$$

This is greater than the correct mission reliability, which is

$$R = P_{11}P_{12}P_{21}P_{22},$$

since mission success is achieved if, and only if, both components function through both phases.

6.3 Methods for the phased mission analysis

6.3.1 Nonrepairable systems

6.3.1.1 Basic event transformation and cut set cancellation

Esary and Ziehms [2] present a method to transform and reduce a phased mission system into an equivalent single phase mission, allowing existing techniques to be applied to obtain mission reliability. In multi-phased mission, the performance of a component in each phase depends on its performance in previous phases. A component will only be working in a phase if it works successfully through all previous phases.

Therefore, a component c in phase j can be replaced by a series system of components which would represent the performance of component c in all phases up to and including the phase j, $c_1, c_2, ..., c_j$. If using fault tree analysis, the single event input of the failure of component c is replaced by an OR combination of the failure of component c in any phase up to and including phase j.

To illustrate the method the following phased mission is considered [3]. The system consists of three non-repairable components A, B and C. The reliability network for this system is given in Figure 6.2.

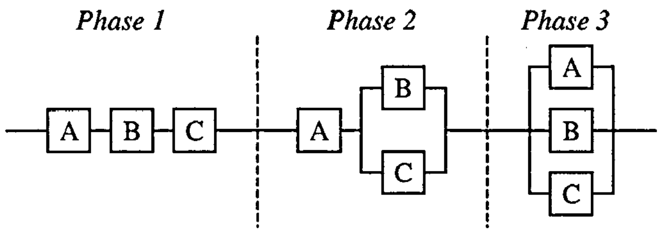


Figure 6.2: Reliability network for the example phased mission system

To accomplish the mission the system must work through all three phases. To accomplish phase 1 all components must work through the phase. If phase 1 is completed successfully, the system enters phase 2, to accomplish this successfully component A, and at least one of the components B and C, must work through the phase. To accomplish phase 3 at least one of the components (A, B or C) must work through the phase.

The fault trees for each phase are represented in Figure 6.3.

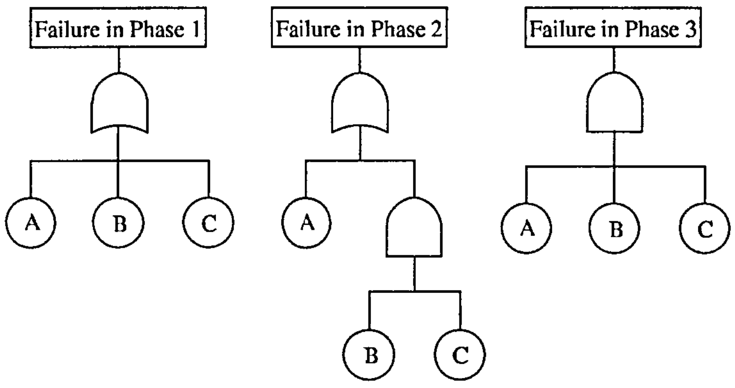


Figure 6.3: Fault tree representation of individual phase failures

Component failure in each phase i is represented by A_i, B_i, C_i. In order to transform the multi-phase mission problem into single-phase mission, all failure events (A, B, C) are replaced by an OR combination of failure events for that and all preceding phases. For example, failure event A in phase 2 will be replaced by OR combination of A_1 (failure of component A in phase 1) and A_2 (failure of component A in phase 2, given that it was functioning through the phase 1). The fault tree for the transformed multi-phase mission problem into a single-phase mission is shown in Figure 6.4.

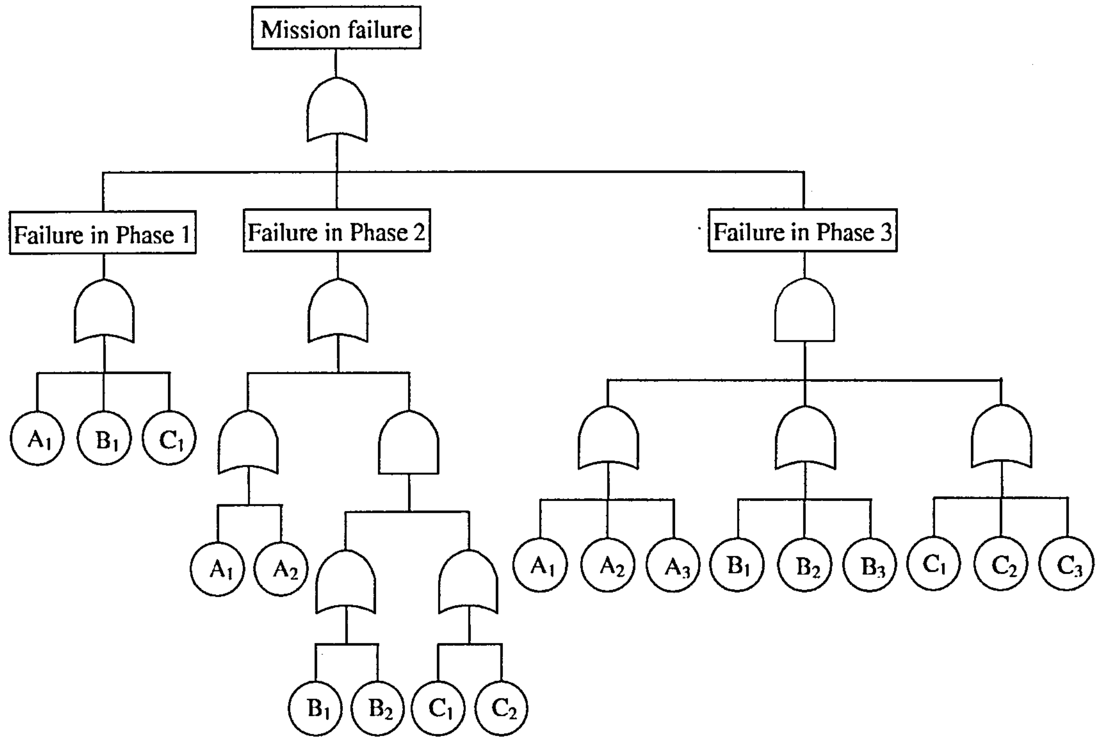


Figure 6.4: Equivalent single-phase mission

The system reliabilities are given by

$$\begin{aligned}
\text{Phase 1 } R_1 &= \rho_{A_1} + \rho_{B_1} + \rho_{C_1} - \rho_{A_1}\rho_{B_1} - \rho_{B_1}\rho_{C_1} - \rho_{A_1}\rho_{C_1} + \rho_{A_1}\rho_{B_1}\rho_{C_1} \\
\text{Phase 2 } R_2 &= \rho_{A_1}\rho_{A_2}\left(\rho_{B_1}\rho_{B_2} + \rho_{C_1}\rho_{C_2} - \rho_{B_1}\rho_{B_2}\rho_{C_1}\rho_{C_2}\right) \qquad (6.1) \\
\text{Phase 3 } R_3 &= \rho_{A_1}\rho_{A_2}\rho_{A_3}\rho_{B_1}\rho_{B_2}\rho_{B_3}\rho_{C_1}\rho_{C_2}\rho_{C_3}
\end{aligned}$$

where ρ_{c_j} is the conditional reliability of component c in phase j:

$$\rho_{c_1} = P\left[x_c(t_1) = 0\right] \text{then, } \rho_{c_j} = P\left[x_c(t_j) = 0 \mid x_c(t_{j-}) = 0\right] \text{ for } j = 2, \ldots, n$$

In order to determine the overall system reliability accurately Esary and Ziehms [2] introduced the concept of cut set cancellation. The rule says, that if the minimal cut sets of an earlier phase contain any minimal cut sets from a later phases, they may be removed from the earlier phase. This can be done as mission failure is the only consideration, and there is no need to repeat such events as later phases take into account the failure of the components in all phases up to the inspected phase.

For the example described earlier (Figure 6.2), the minimal cut sets for each phase are as follow:

Phase 1	Phase 2	Phase 3
A	A	ABC
B	BC	
C		

If component A fails in phase 1 then it will be failed in phase 2. Therefore, cut set A can be removed from the phase 1 as it is a minimal cut set for the later phase as well. In that case the cut sets for the multi-phase mission are as follow:

Phase 1	Phase 2	Phase 3
B	A	ABC
C	BC	

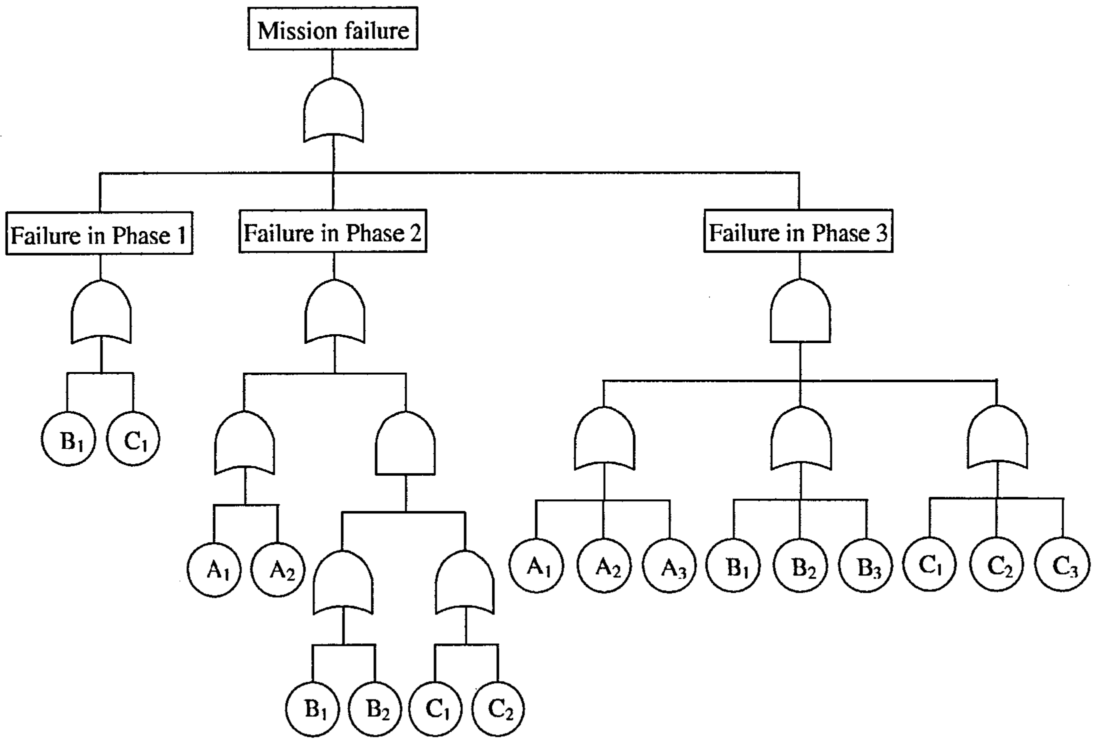


Figure 6.5: Equivalent single-phase mission after cut set cancellation

The method proposed by Esary nd Ziehms is capable of transforming a multi-phase mission into an equivalent single phase mission to allow the use of existing reliability techniques. The cut set cancellation presents a more simple way to transform the system. However, if the cut set cancellation is applied before transformation of the multi-phase mission to the single-phase mission, the fault tree represented in Figure 6.4 would look slightly different (see Figure 6.5). But if cut sets are removed to produce single-phase mission, it becomes impossible to calculate individual phase failure probabilities which may be desirable, note La Band and Andrews [3].

6.3.1.2 Approximate methods for mission unreliability

An important problem of phased mission analysis is to calculate exact or obtain bound for mission unreliability. The work by Esary and Ziehms [2] was reviewed by Burdick at el [1] to suggest methods for obtaining approximate results for mission reliability.

The method suggested by Esary and Ziehms can be applied to the original fault tree of a phased mission system, but the transformation of each basic event C in phase j into a series of events, $c_1, \ldots, c_j$ leads to a large increase in the number of minimal cut sets of the mission. Therefore, it is difficult to calculate the exact mission unreliability. As a solution to this, there are methods developed to estimate the system unreliability without using basic event transformation.

Inclusion-exclusion expansion of phase unreliabilities

The minimal cut sets are obtained for each phase of the original system. The unreliability of phase j, Q_j, is calculated using the inclusion-exclusion equation 2.10 for the minimal cut sets of phase j. The conditional basic event C reliability ρ_{c_j} was obtained in equation 6.1, and the unconditional basic event C reliability p_{c_j} is derived from this in equation:

$$p_{c_j} = P\left[x_c(t_j) = 0\right] = \prod_{i=i}^{j} \rho_{c_i}, \text{ for } j = 1, \ldots, n \tag{6.2}$$

An approximation for mission unreliability $\overline{Q}_{IN-EX}$ can be expressed as a product of the individual phase reliabilities:

$$\overline{Q}_{IN-EX} = \prod_{j=i}^{n} R_j \tag{6.3}$$

In practice, the usual approximation used for mission unreliability is obtained by the sum of individual phase unreliabilities:

$$Q_{IN-EX} \leq \sum_{j-1}^{n} Q_j \tag{6.4}$$

The approximation can also be applied after cut set cancellation to give another approximation for mission unreliability, $Q_{IN-EX(CC)}$. This bound will usually give a result smaller than the one without cut set cancellation due to the fact that there would be fewer cut sets in each phase.

Minimal cut set bound

The minimal cut sets are obtained for each phase from the original logic model. The probability of failure of cut set C_i in phase j is given by

$$q_{c_{ij}} = \prod_{l=i}^{N_{c_{ij}}} P(c_l) \tag{6.5}$$

where c_l is occurrence of basic event c in cut set C_i of phase j, $N_{c_{ij}}$ is the number of basic events in minimal cut set C_i of phase j.

The reliability of phase j is then estimated using minimal bound:

$$R_j = \prod_{i=i}^{N_{mcs_j}} p_{c_{ij}} \tag{6.6}$$

where N_{mcs_j} is number of minimal cut sets in phase j, $p_{c_{ij}}$ is the probability of success of cut set C_i in phase j.

The approximation for the reliability of the mission using minimal cut bound $\overline{Q}_{MCB}$ is obtained the same way as in equation 6.3. This method also can be used after applying the cut set cancellation technique to give approximation of the mission reliability $Q_{MCB(CC)}$.

The approximate methods described above do not account for the outcome of previous phases, therefore these bounds are only estimates. However, such techniques can be useful in finding estimations for systems containing a large number of components where an exact solution would be difficult to calculate or costly.

6.3.1.3 Expected number of failures

A method for calculating the expected number of system failures for a phased mission was developed by Montague and Fussell [30]. They state, that the system expected number of failures is a valuable system reliability characteristic when the system is repairable or non-repairable. According to Montague and Fussell, an expected number of failures much less than unity is desired during the mission for most systems. The proposed method is applicable to both repairable and nonrepairable systems.

The standard method for obtaining top event frequency for a single-phase mission is given in equation 2.14. This is the contribution from the occurrence of at least one minimal cut set minus the contribution of the occurrence of minimal cut sets when the system has already failed. The expected number of system failures is then obtained by the integral of this parameter over a specified time interval in equation 2.20.

This principle is adapted by Montague and Fussell and the expected number of failures for a phased mission with n phases is given by

$$ENF(t_0, t_n) = \sum_{i=1}^{n} \int_{t_{i-1}}^{t_i} W_T(t)dt + \sum_{i=1}^{n-1} \text{boundary contribution} \tag{6.7}$$

The first term in the equation (6.7) is the sum of the expected number of failures occurring during each phase of the mission. The integral term is separated into n phases because the integrand becomes a new function with each new phase. The second term accounts for the TOP event occurring as a phase boundary is crossed. This boundary contribution is needed since it is possible for a combination of basic events to exist at the end of one phase without resulting in the TOP event, but which will cause the occurrence of the TOP in the next phase.

To evaluate this boundary jump, let Δt be an arbitrarily small length of time spanning across the i'th phase boundary. Thus, the expected value of the number of system failures in this Δt time interval

$$\begin{aligned} ENF(i) &= (0 \text{ failures in } \Delta t) \cdot P\left[S\left(t_i - \frac{\Delta t}{2}\right) \cap \overline{S}\left(t_i + \frac{\Delta t}{2}\right)\right] \\ &\quad +(1 \text{ failure in } \Delta t) \cdot P\left[S\left(t_i - \frac{\Delta t}{2}\right) \cap \overline{S}\left(t_i + \frac{\Delta t}{2}\right)\right] \\ &\quad +\text{higher order terms,} \end{aligned} \tag{6.8}$$

where $S\left(t_i - \frac{\Delta t}{2}\right)$ - top event does not exist at time t_i, $\overline{S}\left(t_i + \frac{\Delta t}{2}\right)$ - top event exists at time t_i.

The higher order terms account for the system failing more than one time during the Δt time interval. The failure logic models used to determine

$S\left(t_i - \frac{\Delta t}{2}\right)$ and $\overline{S}\left(t_i + \frac{\Delta t}{2}\right)$ are from phase i and phase $i+1$ respectively. Taking the limit of equation (6.8) as Δt approaches zero, it becomes

$$ENF(i) \quad = P\left[S(t_{i-}) \cap \overline{S}(t_{i+})\right] \tag{6.9}$$

where $S(t_{i-})$ - top event does not exist at the instant before the transition, $\overline{S}(t_{i+})$ - top event exists at the instant after the transition.

Equation (6.9) is the contribution to the mission expected number of failures due only to the logic model changing. Because the transition between phases is assumed to be instantaneous, the state of a basic event does not change during the transition from one phase to the next. Thus, equation (6.9) does not express a basic event changing states that contributes to the TOP event changing states.

With (6.9), the expected number of failures of the TOP event can be expressed as

$$ENF(t_0, t_n) = \sum_{i=1}^{n} \int_{t_{i-1}}^{t_i} W_T(t)dt + \sum_{i=1}^{n-1} P\left[S(t_{i-}) \cap \overline{S}(t_{i+})\right] \tag{6.10}$$

Calculation of the boundary contribution in (6.10), $P\left[S(t_{i-}) \cap \overline{S}(t_{i+})\right]$, requires using the minimal path sets of the failure logic model of one phase and the minimal cut sets of the failure logic model of the next consecutive phase. Using Boolean algebra, an expression for the TOP event not existing at the end of one phase and existing at the beginning of the next can be written in terms of the basic events included in these minimal cut sets and path sets. The method was applied to an emergency core cooling system for a boiling water reactor (see [30]).

6.3.1.4 *Reliability of periodic, coherent, binary systems*

Veatch [33] considers a periodic system without repair for phased mission analysis. In the work it is stated that the single-phase system is useful for approximating the reliability and mean life of the periodic system and it is much more simple to analyze than exact transformations to a single-phase system.

The concept of a binary system is extended to phased missions by considering a separate structure function for each phase of a mission. For s-coherent systems without repair, the system cannot return to a working state from a failed state within a phase. A system is s-coherent if:

1. a component failure cannot cause the system to transmit from failed to working;
2. at least one component is relevant to the state of the system.

Hence, the event that the system functions during phase j can be expressed as $\{\phi_j(X(t_j)) = 1\}$, where $\phi_j(X)$ is a system structure function in phase j. The event that the system functions throughout the mission can be expressed as $\{\phi_j(X(t_j)) = 1, \ldots, \phi_m(X(t_m)) = 1\}$.

Approximate techniques reviewed by Burdick *et al* [1] treat the successful completion of each phase as s-independent events and system reliability is given by multiplying reliability number for each phase. Esary and Ziehms [2] show that using the component reliabilities for phase j gives a lower bound for system reliability and using conditional phase reliabilities gives an upper bound. When cut set cancellation or phase cancellation is applied to these approximations, their accuracy is improved.

6.3.1.4.1 Lower bound systems and periodic systems

Another technique [33] that can be used to approximate phased-mission reliability is to construct a lower bound single-phased system, defined by:

$$\phi_{LB}(X) = \begin{cases} 1, & if\ \phi_j(X) = 1,\ for\ j = 1, \ldots, m \\ 0, & otherwise. \end{cases} \tag{6.11}$$

Algebraically, ϕ_{LB} can be computed as

$$\phi_{LB}(X) = \prod_{j=1}^{m} \phi_j(X). \tag{6.12}$$

The block diagram for the lower bound system is constructed by placing the block diagram for each phase in series. The concept of cut set cancellation can be used. However, for the lower bound system, cancellation can be done in earlier or later phases[33]. Hence, a cut set that contains a cut set from any other phase can be cancelled regardless of sequences.

The lower bound system is particularly valuable in analyzing the performance of a system that repeatedly performs the same mission without repair. If the structure function is viewed as a function of time, it is periodic with period $L = t_m$ for each system state X. Such a system will be called periodic. The results which follow can easily be extended to periodic systems with continuously varying structure functions, instead of discrete phases, states [33].

6.3.1.4.2 Reliability bounds for periodic systems

The reliability of a s-coherent periodic system with period L is related to that of the lower bound system by:

$$T_{LB} \leq T \leq T_{LB} + L \tag{6.13}$$

where T is system life (time at which system fails), T_{LB} - life of the lower bound system.

The usefulness of (6.13) for establishing reliability bounds is shown by [33] and is given in (6.14) and (6.15).

$$R_{LB}(t) \leq R(t) \leq R_{LB}(t-L) \tag{6.14}$$

The lower bound may be useful after one mission ($t = L$). Both bounds are restrictive for a large number of missions without repair, notices [33]. The mean life of a periodic coherent system can be bounded by the mean life of the lower bound system:

$$E\left[T_{LB}\right] \leq E\left[T\right] \leq E\left[T_{LB}\right] + L. \tag{6.15}$$

The bounds in (6.15) are tight if $E[T] \gg L$. For complex systems, mean life often must be computed using numerical integration of the reliability function. In this case, using R_{LB} instead of R becomes particularly important computationally.

6.3.1.5 Generalized intersection and union concept

Dazhi and Xiaozhong [34] propose a different method to obtain estimates of system unreliabilities in different phases as well as mission unreliability. The method does not need basic event transformation. In the paper they present a generalized intersection and union concept that could be used to investigate the advantages and limits of various approximation techniques and indicate ways of improvement. The assumptions of the proposed method are as follow:

1. Logic model contains non-repairable basic events.
2. Logic model is coherent.
3. Basic events are statistically independent (s-independent) in a failure.

4. Transition time between any two successive phases is instantaneous.

A^j is used to denote that component A failed in phase j and worked in all previous phases and $A^{(j)}$ is used to denote that component A is failed in phase j: the component A is failed in phase j if it failed in phase j or any of the previous $j-1$ phases.

6.3.1.5.1 *Generalized intersection and union concept*

Boolean algebra is used as the foundation of fault tree analysis. But for phased mission problems the initial condition of each phase and the relationship of the basic events in different phases should be taken into account. In [34] in the fault tree for phase j, the basic event $A^{(j)}$ is transformed to a series logic of j basic events, $A^1 + A^2 + \ldots + A^j$.

Suppose that $j \geq k \geq 1$. Then

$$\begin{aligned} A^{(j)} &= A^1 \cup A^2 \cup \ldots \cup A^j = \bigcup_{i=1}^{k} A^i \bigcup_{i=k+1}^{j} A^i = A^{(k)} \bigcup_{i=k+1}^{j} A^i \\ A^{(k)} \cap A^{(j)} &= A^{(k)} \cap \left(A^{(k)} \bigcup_{i=k+1}^{j} A^i \right) = A^{(k)} \bigcup_{i=k+1}^{j} \left(A^{(k)} A^i \right) = A^{(k)} \qquad (6.16) \\ A^{(k)} \cup A^{(j)} &= A^{(k)} \cup \left(A^{(k)} \bigcup_{i=k+1}^{j} A^i \right) = A^{(k)} \bigcup_{i=k+1}^{j} A^i = A^{(j)} \qquad (6.17) \end{aligned}$$

Here the intersection and the union concept is extended to events in different phases.

In [34] equations (6.16) and (6.17) are added to the list of Boolean algebra principles to consider the time-dependent effect between cut sets in different phases. The same basic event in different phases is considered to be a different event.

6.3.1.5.2 *Inclusion-exclusion principle*

The inclusion-exclusion principle is a method that provides successive upper and lower bounds on system unreliability and converge to the exact unreliability by considering terms to account for intersections of cut-sets.

$$\begin{aligned} P\left(\bigcup_{i=1}^{n} C_i\right) &= \sum_{i=1}^{n} P(C_i) - \sum_{i \neq j} P(C_i \cap C_j) + \sum_{i \neq j \neq k} P(C_i \cap C_j \cap C_k) \\ &\quad - \ldots (-1)^{n-1} P(C_1 \cap \ldots \cap C_n) \qquad (6.18) \end{aligned}$$

C_i, $1 \leq i \leq n$ can be a basic event or a minimal cut set.

By corporation of the generalized intersection and union concept discussed above, the inclusion-exclusion principle can be used directly to solve the phased mission problem. For a phased mission problem, C_1, C_2, ..., C_n in equation (6.18) may be basic events or minimal cut sets for different phases.

6.3.1.5.3 Methodology of mission unreliability calculation

In a phased mission problem, the system is failed in phase n if it has failed in phase n or any of the previous $n - 1$ phases. This can be expressed as

$$X^{(n)} = X^1 \cup X^2 \cup X^3 \cup \ldots \cup X^n \tag{6.19}$$

where $X^{(n)}$ is the event that the system is failed in phase n. X^i is the event that the system fails for the first time in phase i.

The system mission unreliability Q_s can be calculated by

$$Q_s = P(X^{(n)}) = P\left(\bigcup_{i=1}^{n} X^i\right) = P\left[\bigcup_{i=1}^{n}\left(\bigcup_{j=1}^{m_i} C_j^{(i)}\right)\right] \tag{6.20}$$

where $X^{(i)} = \bigcup_{j=1}^{m_i} C_j^{(i)}$, $C_j^{(i)}$ is a minimal cut set for X^i, m_i is the number of minimal cut sets in phase i, and n is the number of phases.

When generalized union concept is used in equation (6.20), the mission cut sets cancellation can be realized automatically [34].

6.3.1.6 Method of Lee and Hong

Lee and Hong [35] note that methods based on minimal cut set analysis are not very efficient. As the number of phases increases, they require a very complicated and time consuming procedure. Hence, the calculation of the exact unreliability of a mission is usually expensive. Lee and Hong give a closed form mathematical expression of a phased mission system reliability. They consider a system where the failure rate of a component and the number of added redundancies change during the mission. The mission consists of N time phases. In phase k the probability of component failure and number of redundancy added at the beginning of the phase are given as q_k and x_k respectively. All components in redundancy are assumed to operate whenever possible.

Assume that at the beginning of phase k there are r_k components. r_k consists of two parts: one comes from survival components of previous phases, and the other from added redundancy, x_k of current phase. r_k can have any value between $1 + x_k$ and $1 + \sum_{i=1}^{k-1} x_i + x_k$.

Let $\bar{r}_k$ be the number of components remaining at the end of phase k. Then the transition probability that there are j component at the end of phase k starting from i components at the beginning of the phase k is given by

$$P^k_{i,j} = P(\bar{r}_k = j | r_k = i) = \begin{cases} \begin{pmatrix} i \\ j \end{pmatrix} q_k^{i-j}(1-q_k)^j & , \text{if } i \geq j \\ 0 & , \text{otherwise} \end{cases}$$

The transition probability matrix A^k of the number of components in phase k can be given as

$$\begin{aligned} A^k &= \left(P^k_{i,j}\right), \text{ where } 1 + x_k \leq i \leq 1 + \alpha_k,\ 0 \leq j \leq 1 + \alpha_k \\ \alpha_k &= \sum_{i=1}^{k} x_i \end{aligned}$$

The phased mission reliability can be expressed as follows

$$\begin{aligned} R_s = {} & 1 - \{P[\text{fail in phase 1}] + P[\text{OK in phase 1 } \textit{and} \text{ fail in phase 2}] + \ldots \\ & + P[\text{OK in phases 1 to } N-1 \textit{ and} \text{ fail in phase } N]\} \end{aligned}$$

Each term of the above equation can be obtained by multiplication of transition probabilities.

$$\begin{aligned} P[\text{fail in phase 1}] &= P^1_{1+x_1,0} = F_1 \\ P[\text{OK in phase 1 } \textit{and} \text{ fail in phase 2}] &= \sum_{i_1}^{1+\alpha_1} P^1_{1+x_1,i_1} P^2_{i_1+x_2,0} = F_2 \\ &\vdots \end{aligned}$$

$$\begin{aligned} & P[\text{OK in phases 1 to } N-1 \textit{ and} \text{ fail in phase } N] \\ & = \sum_{i_{N-1}}^{1+\alpha_{N-1}} \cdots \sum_{i_2}^{1+\alpha_2} \sum_{i_1}^{1+\alpha_1} P^1_{1+x_1,i_1} P^2_{1+x_2,i_2} \cdots P^N_{i_{N-1}+x_N,0} \\ & = F_N \end{aligned}$$

Finally,

$$\begin{aligned} R_s &= 1 - \left\{ P^1_{1+x_1,0} + \sum_{i_1}^{1+\alpha_1} P^1_{1+x_1,i_1} P^2_{i_1+x_2,0} + \ldots \right. \\ &\quad \left. + \sum_{i_{N-1}}^{1+\alpha_{N-1}} \cdots \sum_{i_2}^{1+\alpha_2} \sum_{i_1}^{1+\alpha_1} P^1_{1+x_1,i_1} P^2_{1+x_2,i_2} \cdots P^N_{i_{N-1}+x_N,0} \right\} \\ &= 1 - \{F_1 + F_2 + \ldots + F_N\} \end{aligned}$$

6.3.1.7 *Phased mission system analysis using boolean algebraic methods*

Somani and Trivedi [29] describe a technique for phased mission system reliability analysis based on Boolean algebraic methods. They develop a phase algebra (see section 6.3.1.7.1) to account for the effects of variable configurations and success criteria from phase to phase. They do not create a single-phase mission, but handle one phase at a time and compute the overall unreliability of the entire mission.

Somani and Trivedi give four possible cases which may occur at the time of a phase transition from phase i to phase $i+1$:

1. A combination of component failures does not lead to system failure in both phases i and $i+1$.

2. A combination of component failures leads to system failure in both phases i and $i+1$.

3. A combination of component failures does not imply system failure in phase i but is treated as system failure in phase $i+1$.

4. A combination of failures implies system failure in phase i but does not imply system failure in phase $i+1$.

6.3.1.7.1 *Phase algebra*

Using the technique described by [29] the example shown in Figure 6.2 earlier has been considered in detail.

Let $A = 1$ mean that component A has failed. Then $\overline{A} = 0$ says that component A has failed and $\overline{A} = 1$ means that component A is operational. Using this notation for the example of phased mission system depicted in the Figure 6.2 (see section 6.3.1.1), the following Boolean expression describe the failure combinations for phases 1, 2 and 3.

$$\begin{aligned} E_1 &= A+B+C \\ E_2 &= A+BC \\ E_3 &= ABC \end{aligned}$$

Let $\overline{A}_i$ denote the event that component A is operational during the interval from the start of the mission until the end of the phase i. This automatically implies that the component is operational during earlier phases as well.

Let i and j be two phases and let $i < j$. The rules in Table 6.1, given by Somani and Trivedi [29], should be used to simplify the logic expressions.

Table 6.1: Combining rules

$\overline{A}_i\overline{A}_j \rightarrow \overline{A}_j$	$A_i + A_j \rightarrow A_i$
$A_iA_j \rightarrow A_i$	$\overline{A}_i + \overline{A}_j \rightarrow \overline{A}_j$
$A_i\overline{A}_j \rightarrow 0$	$\overline{A}_i + A_j \rightarrow 1$

The first combination in Table 6.1 ($\overline{A}_i\overline{A}_j$) means that component A was working until the end of phase i and until the end of phase j which is equivalent to component A working until the end of phase j as this automatically implies that component was operational during earlier phases.

$\overline{A}_iA_j$ and $A_i + \overline{A}_j$ cannot be simplified any further. What the first combination ($\overline{A}_iA_j$) means is that component A is operational until the end of phase i and then fails sometime between the end of phase i and the end of phase j. The second term has no physical meaning. Also, it is not possible for a component fails during a phase and then be operational during a later phase ($A_i\overline{A}_j$). Hence $A_i\overline{A}_j \rightarrow 0$.

6.3.1.7.2 *Example*

For the example considered the system has three components and there are three phases: all components must work through the phase 1 for it to be successful. If phase 1 is accomplished the system enters phase 2, to complete this component A and component B or C must work, to complete phase 3 all components A, B and

C must not fail. The failure combinations of phases 1, 2 and 3 are defined by E_1, E_2 and E_3, respectively.

Then phase failure combinations for the phase i (PFC_i), which are treated as success combinations for all subsequent phases are given by

$$PFC_i = (\ldots((E_i \cdot \overline{E}_{i+1}) \cdot \overline{E}_{i+2}) \ldots \cdot \overline{E}_n)$$

In the above expression, only those combinations are included which are failure combinations in phase i but are not failure combinations in any of subsequent phases [29]. This expression can be simplified as

$$PFC_i = E_i \cdot (\overline{E_{i+1} + \ldots + E_n})$$

Then for the phase 1 we have

$$\begin{aligned} PFC_1 &= (E_1 \cdot \overline{E}_2) \cdot \overline{E}_3 \\ &= ((A_1 + B_1 + C_1) \cdot (\overline{A_2 + B_2C_2}) \cdot (\overline{A_3B_3C_3}) \\ &= \overline{A}_3\overline{B}_2C_1 + \overline{A}_3B_1\overline{C}_2 + \overline{A}_2\overline{B}_3C_1 + \overline{A}_2B_1\overline{C}_3 \end{aligned}$$

And for the phase 2 (PFC_2):

$$\begin{aligned} PFC_2 &= E_2 \cdot \overline{E}_3 \\ &= (A_2 + B_2C_2) \cdot (\overline{A_3B_3C_3}) \\ &= \overline{A}_3B_2C_2 + A_2\overline{B}_3 + A_2\overline{C}_3 \end{aligned}$$

General formula for system unreliability is

$$Q_{sys} = P(E_n) + \sum_{i=1}^{p-1} P(PFC_i),$$

where $P(E_n)$ is the probability of failure of the last phase (phase n), $P(PFC_i)$ is the probability of phase failure combinations for phase i.

Then the system unreliability for this example is given by [29]

$$Q_{sys} = P(E_3) + P(PFC_1) + P(PFC_2).$$

$P(E_3)$ in the equation above is

$$P(E_3) = P(A_3B_3C_3)$$

The other two terms, PFC_1 and PFC_2, are computed as follows [29]:

$$\begin{aligned}
P(PFC_1) &= P(\overline{A}_3\overline{B}_2C_1+\overline{A}_3B_1\overline{C}_2+\overline{A}_2\overline{B}_3C_1+\overline{A}_2B_1\overline{C}_3)\\
&= P(\overline{A}_3\overline{B}_2C_1)+P((\overline{A}_3B_1\overline{C}_2+\overline{A}_2\overline{B}_3C_1+\overline{A}_2B_1\overline{C}_3)(\overline{\overline{A}_3\overline{B}_2C_1}))\\
&= P(\overline{A}_3\overline{B}_3C_1)+P((\overline{A}_3B_1\overline{C}_2+\overline{A}_2\overline{B}_3C_1+\overline{A}_2B_1\overline{C}_3)(A_3+B_2+\overline{C}_1))\\
&= P(\overline{A}_3\overline{B}_3C_1)+P(\overline{A}_3B_1\overline{C}_2+\overline{A}_2A_3\overline{B}_3C_1+\overline{A}_2B_1\overline{C}_3)\\
&= P(\overline{A}_3\overline{B}_3C_1)+P(\overline{A}_3B_1\overline{C}_2)\\
&\quad +P((\overline{A}_2A_3\overline{B}_3C_1+\overline{A}_2B_1\overline{C}_3)(A_3+\overline{B}_1+C_2))\\
&= P(\overline{A}_3\overline{B}_3C_1)+P(\overline{A}_2A_3\overline{B}_3C_1+\overline{A}_2A_3B_1\overline{C}_3)\\
&= P(\overline{A}_3\overline{B}_3C_1)+P(\overline{A}_2A_3\overline{B}_3C_1)+P((\overline{A}_2A_3B_1\overline{C}_3)(A_2+\overline{A}_3+B_3+\overline{C}_1)\\
&= P(\overline{A}_3\overline{B}_3C_1)+P(\overline{A}_2A_3\overline{B}_3C_1)+P((\overline{A}_2A_3B_1\overline{C}_3)
\end{aligned}$$

$$\begin{aligned}
P(PFC_2) &= P(\overline{A}_3B_2C_2+A_2\overline{B}_3+A_2\overline{C}_3)\\
&= P(\overline{A}_3B_2C_2)+P((A_2\overline{B}_3+A_2\overline{C}_3)(A_3+\overline{B}_2+\overline{C}_2))\\
&= P(\overline{A}_3B_2C_2)+P(A_2\overline{B}_3+A_2\overline{C}_3)\\
&= P(\overline{A}_3B_2C_2)+P(A_2\overline{B}_3)+P(A_2\overline{C}_3(\overline{A}_2+B_3)\\
&= P(\overline{A}_3B_2C_2)+P(A_2\overline{B}_3)+P(A_2B_3\overline{C}_3)
\end{aligned}$$

So, the system unreliability is equal to

$$\begin{aligned}
Q_{sys} &= P(A_3B_3C_3)+P(\overline{A}_3\overline{B}_3C_1)+P(\overline{A}_2A_3\overline{B}_3C_1)+\\
&+ P((\overline{A}_2A_3B_1\overline{C}_3)+P(\overline{A}_3B_2C_2)+P(A_2\overline{B}_3)+P(A_2B_3\overline{C}_3).
\end{aligned}$$

6.3.1.7.3 *Sum of disjoint products and its phased-extension*

Ma and Trivedi [28] introduce the sum of disjoint phase products (SDPP), which is a phased-extension of the sum of disjoint products (SDP) formula.

The sum of disjoint products formula is one of the techniques that is used to compute the probability of a union of a set of events in a single-phased system [28].

Let E_i be the event that all the components in the minimal cut set MC_i fail: the event E_i is a Boolean expression describing a single minimal cut set MC_i. The SDP formula for calculating the unreliability of the system is:

$$Q_S = P\left[\bigcup_{i=1}^{n} E_i\right] = P\left[E_1 \cup (\overline{E}_1E_2) \cup (\overline{E}_1\overline{E}_2E_3) \cup \ldots \cup (\overline{E}_1\overline{E}_2\ldots\overline{E}_{n-1}E_n)\right], \quad (6.21)$$

where n is the total number of minimal cut sets. Define the constituent $CS_1 = E_1$ and in general, $CS_i = \overline{E}_1\overline{E}_2 \ldots \overline{E}_{i-1}E_i$, where $1 \leq i \leq n$. Since the constituents CS_i in equation (6.21) are disjoint from each other, the final SDP formula for calculating the unreliability of the system is:

$$Q_s = \sum_{i=1}^{n} P(CS_i) \tag{6.22}$$

The most important thing of the SDP formula is to obtain the disjoint constituent CS_i, for $i > 1$.

To calculate the unreliability of the phased mission system, the sum of disjoint products formula was extended into the sum of disjoint phased products (SDPP) formula, [28].

Let PE_1 be the event that a phase mission system is failed in phase i. The SDPP formula for the unreliability of the phased mission system is:

$$\begin{aligned} P_{PMS} &= P\left[\bigcup_{i=1}^{p} PE_i\right] \qquad (6.23) \\ &= P\left[PE_1 \cup (\overline{PE}_1 PE_2) \cup (\overline{PE}_1\overline{PE}_2 PE_3) \cup \ldots \cup (\overline{PE}_1\overline{PE}_2 \ldots \overline{PE}_{p-1} PE_p)\right] \end{aligned}$$

where p is the total number of phases for the phased mission system. In equation (6.21) event E_i represents one single minimal cut set. In equation (6.23) event PE_i represents a set of minimal cut sets, in which the minimal cut sets are generally non-disjoint [28]. The complement of PE_i is normally a set of non-disjoint phase products as well. Define the phase constituent $PC_1 = PE_1$ and, in general, $PC_i = \overline{PE}_1\overline{PE}_2 \ldots \overline{PE}_{i-1}PE_i$, where $1 < i \leq p$. Generally, the phase products in each PC_i are non-disjoint. If the phase products in a PC_i are mutually disjoint, the PC_i is defined as a disjoint phase constituent, denoted by DPC_i. One of the challenges in using the SDPP formula is to change the PC_i into DPC_i, see [28]. Once the DPC_i are found, the final SDPP formula for calculating the unreliability of the phased mission system is:

$$P_{PMS} = \sum_{i=1}^{p} P(DPC_i). \tag{6.24}$$

6.3.1.8 A BDD-based algorithm for reliability analysis of phased mission systems

Zang, Sun and Trivedi [4] proposed a different algorithm based on binary decision diagrams (BDD) to analyse reliability of phased mission systems. The algorithm uses phase algebra (see section 6.3.1.7.1) to deal with the dependencies across the phases. The theory of binary decision diagrams for single-phased systems is described in Chapter 3.

The relations in the phase algebra in Table 6.1 are different from the ordinary logic relations. A special BDD operation, phase-dependent operation, is derived for these relations. Because BDD structures depend strongly on the order of variables, there are two classes of phase-dependent operation (PDO) [4]:

1. Forward PDO: the order of variables is the same as the phase order - c_1, c_2, $\ldots$, c_n.

2. Backward PDO: the order of variables is the reverse of the phase order - c_n, c_{n-1}, $\ldots$, c_1.

Let $i < j$, and let component C be used in both phases i and j. Using *ite* format, E_i and E_j, when expanded with regard to c_i and c_j, respectively, can be written as:

$$\begin{aligned} E_i &= ite[c_i, (E_i)_{c_i=1}, (E_i)_{c_i=0}] = ite[c_i, G_1, G_2] \\ E_j &= ite[c_j, (E_j)_{c_j=1}, (E_j)_{c_j=0}] = ite[c_j, H_1, H_2] \end{aligned}$$

Zang *et al* [4] give two lemmas.

Lemma 1: For the forward PDO,

$$ite(c_i, G_1, G_2) \diamond ite(c_j, H_1, H_2) = ite(c_i, G_i \diamond H_1, G_2 \diamond E_j) \tag{6.25}$$

Lemma 2: For the backward PDO,

$$ite(c_i, G_1, G_2) \diamond ite(c_j, H_1, H_2) = ite(c_i, E_i \diamond H_1, G_2 \diamond H_2) \tag{6.26}$$

6.3.1.8.1 BDD algorithm for phased mission system

The main procedure of reliability analysis of phased mission system using BDD is as follows [4]:

1. To obtain the failure function for each variable use:

$$F_{c_j}(t) = \left[1 - \prod_{i=1}^{j-1}(1 - p_{c_i}(T_i))\right] + \left[\prod_{i=1}^{j-1}(1 - p_{c_i}(T_i))\right] \cdot p_{c_j}(t);$$

the time t is measured from the beginning of phase j so that $0 \leq t \leq T_j$, p_{c_i} is the failure function of mini component c_i, T_i - duration of phase i. The first term in the above expression represents the probability that component C has already failed in the previous phases. The second term represents the probability distribution of lifetime of the component in phase j.

2. Order components and their corresponding variables using the following heuristics: *Weights with a value of 1 are assigned to each leaf of the fault tree. The weight of each gate is obtained by adding the weights of its inputs. When the weights are known in the whole tree, a depth-first traversal of the tree is made, choosing at each level the sons of a gate by order of increasing weights. During this traversal, the variables are put in the ordered list as soon as they are encountered.*

3. Generate the BDD for each phase using ordinary logical operations.

4. Use the phase algebra and the corresponding backward PDO to combine these BDD to obtain the final BDD from the BDD of each phase. When backward phase-dependent operations are used to generate a BDD for phased mission systems, the cancellation of common components can be done automatically during the generation of the BDD without any additional operations.

5. Calculate the unreliability of phased mission system from the final BDD. The calculation of the system unreliability from the final BDD is easy and fast as the BDD is based on Shannon decomposition.

To illustrate the method consider example shown in Figure 6.3. The equivalent system for the mission (single phase) is shown in Figure 6.4. The final BDD for this example is shown in Figure 6.6. The *ite* structure for mission failure could be expressed as:

$$\begin{aligned} MISSION\ FAILURE &= ite(A_3, ite(A2, 1, ite(B_3, ite(B_1, 1, ite(C_3, 1, 0)), \\ &= ite(C_1, 1, 0)), ite(B_2, ite(B_1, 1, iteC_2, 1, 0), ite(C_1, 1, 0)) \end{aligned}$$

6.3.1.9 Imperfect coverage

Xing and Dugan [36] consider the problem of analysing generalized phased mission systems which have combinatorial phase requirements and imperfect

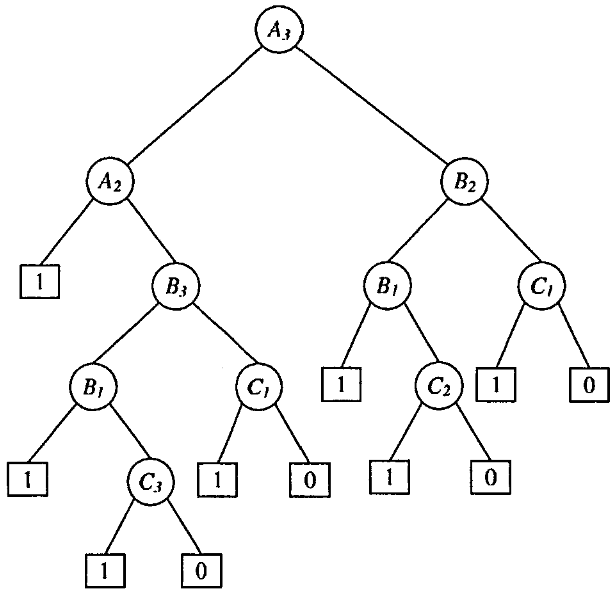


Figure 6.6: BDD representation for the fault tree shown in Figure 6.4

coverage[1]. They note that the phased mission systems are mostly designed for OR-ed phases, which means that if the system fails during any one phase, it fails to achieve the mission. Thus, the reliability of conventional phase-OR phased mission system is the probability that the mission successfully achieves the objectives in all phases. Xing and Dugan notice that there are phased mission systems that have combinatorial phase requirements, which means that a phased mission system could have a failure criterion as any logical combination of the phase failures in terms of phase-AND[2] , phase-K/M[3], and phase-OR[4]. In addition, there exists systems that have more than just binary (success or failure) outcome. Xing and Dugan propose a generalized phased mission system analysis that can incorporate combinatorial phase requirements, multiple grade-level performance criteria and imperfect coverage together. The methodology integrates several methodologies for separate analysis

[1]Imperfect coverage means that single-point failure(uncovered failure) can bring down the entire system despite the presence of fault-tolerance mechanisms

[2]The mission has successfully achieve objectives in all phases - if the system fails during any one phase, it fails to achieve the mission

[3]The mission has successfully achieve objectives in M-K out of M phases - if the system fails during K phases, it fails to achieve the mission

[4]The mission has successfully achieve objectives in at least one phase - if the system fails during all phases, it fails to achieve the mission

of phased missions and imperfect coverage.

The algorithm of Zang, Sun and Trivedi [4] (see section 6.3.1.8) is used by Xing and Dugan to incorporate imperfect coverage as the generalized phased mission systems considered by Xing and Dugan involve imperfect coverage. The basic event transformation (see section 6.3.1.1) deals with the s-dependence across the phases and makes the approach of Zang, Sun and Trivedi possible . To use the algorithm by Zang, Sun and Trivedi first of all basic event transformations have to be applied to the system.

Xing and Dugan consider the phased mission systems that have: specified combinatorial phase requirements, imperfect coverage, and/or multiple grade-level performance criteria. These phased missions are generalized phased mission systems. The conventional phase-OR phased mission system is a special case.

The problem assumed by Xing and Dugan is to derive an exact analytic approach to evaluating the reliability and/or performance (multilevel reliability) of a generalized phased mission system, given as inputs:

a) the combinatorial phase requirements and/or mission performance criteria

b) the duration of phase i: T_i

c) failure distribution for component C_A in phase i: for C_{a_i}, which is conditioned on success of $C_{a_{i-1}}$

d) coverage parameter: r_{a_i}, c_{a_i} and s_{a_i} for each component in each phase

e) failure criteria for each phase

f) mission time.

Imperfect coverage modelling. Computer-based systems usually exhibit multiple failure modes: covered and uncovered failures. In addition, different failure modes have distinct effects on the system failure.

- 'Covered failure' is local to the affected component, it might or might not lead to system failure - depending on the remaining redundancy.
- 'Uncovered failure' is globally malicious, causing immediate system failure.

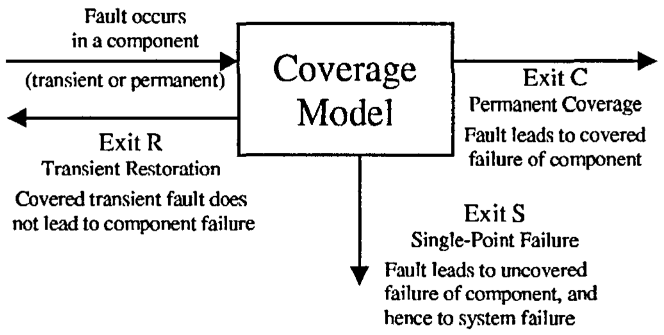

Fault occurs
in a component
(transient or permanent)
Coverage Model
Exit C
Permanent Coverage
Fault leads to covered failure of component
Exit R
Transient Restoration
Covered transient fault does not lead to component failure
Exit S
Single-Point Failure
Fault leads to uncovered failure of component, and hence to system failure

Figure 6.7: General structure of a coverage model

The general structure of imperfect coverage is shown in Figure 6.7. The entry point to the model signifies the occurrence of the fault, and the three exits represent three possible outcomes. If the offending fault is transient, and can be handled without discarding any component, then the transient restoration exit (labelled R) is taken. The permanent coverage exit (labelled C) denotes the determination of the permanent nature of the fault, and the successful isolation and removal of the faulty component. If the permanent coverage exit is reached, then a covered component failure occurs. When a single fault (by itself) causes the system to crash, the single-point failure exit (labelled S) is reached, then an uncovered component failure occurs.

6.3.1.10 Other methods

There are many more methods suggested for the analysis of non-repairable phased mission systems. Some of the methods not mentioned above are based on Markov analysis. There are two possible approaches to the multi-phased system using Markov methods - either to treat each phase individually, or analyse the entire mission with the single model. If the phases of the mission are treated separately, each individual Markov model must be solved separately and then all of them have to be linked by a state probability vector. The alternative approach involves solving a single model for the mission with state space at least equal to the size of the sum of the components in each individual phase model. The problem of constructing a single Markov model for a phase mission system is considered by Dugan [41]. However, the Markov model does suffer from a state explosion problem as the number of components and phases in the mission increases.

Some missions are required to achieve more than one objective. This is investigated by Pedar and Sarma [31] as they consider a transport aircraft with mission objectives as 1) fuel efficiency, 2) no diversion, and 3) no fatalities. The method involves obtaining minimal cut sets for each objective. The probabilities for 5 levels (i.e., low fuel consumption, diversion, no fatalities) of accomplishment of phased mission are calculated.

Burdick *et al* [1, 27] applied method proposed Esary and Ziehms [2] to a typical phased mission problem that may arise in the nuclear power industry. The technique was applied to an emergency core cooling system for a boiling water reactor. The system consisted of 8 subsystems which where considered as components for the analysis. One mission of the emergency core cooling system is to prevent excessive heating of the fuel rods within the reactor vessel as soon as possible after large loss of coolant accident and then keep water circulating to and from the reactor vessel until the rods are cool. After loss of coolant accident has occurred, three phases for the emergency core cooling system were considered: initial core cooling, suppression pool cooling and residual heat removal. Both exact and approximate mission unreliability were calculated. The approximate methods used in the example produced results that were very close to the exact one (difference of 0.2%).

6.3.2 Repairable systems

Clarotti *et al* [37] state that if a system is composed of repairable components, then only an upper bound can be found using the fault tree approach. They notice that in order to find an exact solution to the problem the Markov approach can be successfully used. Clarotti *et al* note that fault tree technique gives an exact result only if a complete independence among system components can be assumed.

Each phase can be identified by: phase number, time interval, system configuration, parameter of interest (reliability, availability), maintainability policy (single, double, ..., multiple).

If no maintenance is provided, and common failures are not considered, then complete independence holds in each phase and the approach based on fault tree technique leads to the exact problem solution. This case was intensively treated by Esary and Ziehms [2].

If reliability with maintenance is of interest, independence cannot be assumed, in the sense that any component may be repaired or not depending upon the system state. Is such cases, methods based on combinatorial reliability cannot be used to find an exact solution; this is the case of the fault tree analysis.[37]

In order to find an exact solution, the method which is not constrained by the independence assumption is needed, such as the analytical Markov approach, state Clarotti *et al.*

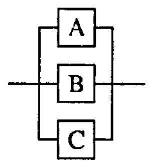
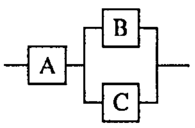
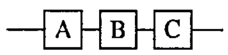
Approach for solving repairable phased missions using analytical Markov method where the mission phase change times are deterministic was discussed by Clarotti *et al* [37] and by Alam and Al-Saggaf [38]. Alam and Al-Saggaf extend the method for the case where mission phase change times are stochastic and present two possible solutions. Smotherman [39, 40] suggests non-homogeneous Markov model saying that it greatly increases modelling flexibility and scope of practical application. Dugan [41] propose discrete-state continuous time Markov model, where all the phases are combined into one model.

6.3.2.1 Markov approach for reliability evaluation

6.3.2.1.1 Deterministic mission phase change time

To illustrate the Markov approach a 3-phase mission is considered, the whole mission profile is shown in Table 6.2 [38].

Table 6.2: Mission profile

Phase No.	Time interval	System configuration (Block diagram)	System task	Parameter of interest	Maintenance policy
1	$(0,t_1)$		x	Reliability	Multiple
2	(t_1,t_2)		y	Reliability	Multiple
3	(t_2,t_3)		z	Reliability	Multiple

That the mission reliability cannot be obtained by simply multiplying the system reliabilities of the various phases was noted by Esary and Ziehms [2]. This is due to the fact that at times at which the system changes its configuration, it must occupy a state which is successful for both the phases involved. So, the system evolution must be such that: during each phase the system can evolve through all the states allowed for that phase. At any phase change time the system must occupy one of the states that are good for both phases, starting from which it begins the evolution in the following phase with the same constraints. The states of the phases are depicted in Table 6.3. State s_1 in Table 6.3 means that all three components are working (1) and it is a success situation for all tree phases. State s_5 means that components A and B are working, but C is failed. This is a success state for phase 1 and phase 2, but a failure state for phase 3. ϕ_i is the structure vector for phase i.

Table 6.3: Phases states description

				Phases		
State	A	B	C	ϕ_1	ϕ_2	ϕ_3
s_1	1	1	1	1	1	1
s_2	0	1	1	1	0	0
s_3	1	0	1	1	1	0
s_4	0	0	1	1	0	0
s_5	1	1	0	1	1	0
s_6	0	1	0	1	0	0
s_7	1	0	0	1	0	0
s_8	0	0	0	0	0	0

From [37] considering the three phases:

a) *First phase: from* 0 *to* t_1. In order to have success during the first phase, the system has not to pass through state s_8. In addition, at the phase change time t_1, it has to occupy one of the states s_1, s_3 or s_5 which are success states for phases 1 and 2.

The state transition rate matrix Λ_1 for phase 1 can be easily obtained using Table 6.3 where state s_8 is an absorbing state and the repair starts as soon as a component fails [38]

$$\Lambda_1 = \begin{bmatrix} -\Sigma_1 & \mu_A & \mu_B & 0 & \mu_C & 0 & 0 & 0 \\ \lambda_A & -\Sigma_2 & 0 & \mu_B & 0 & \mu_C & 0 & 0 \\ \lambda_B & 0 & -\Sigma_3 & \mu_A & 0 & 0 & \mu_C & 0 \\ 0 & \lambda_B & \lambda_A & -\Sigma_4 & 0 & 0 & 0 & 0 \\ \lambda_C & 0 & 0 & 0 & -\Sigma_5 & \mu_A & \mu_B & 0 \\ 0 & \lambda_C & 0 & 0 & \lambda_A & -\Sigma_6 & 0 & 0 \\ 0 & 0 & \lambda_C & 0 & \lambda_B & 0 & -\Sigma_7 & 0 \\ 0 & 0 & 0 & \lambda_C & 0 & \lambda_B & \lambda_A & 0 \end{bmatrix} \quad (6.27)$$

where λ_i is failure rate, μ_i - repair rate, Σ_i - sum of the entries of the i-th column.

Therefore,

$$\dot{\mathbf{P}}(t) = \Lambda_1 \mathbf{P}(t) \quad (6.28)$$

with initial conditions

$$\mathbf{P}(0) = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T \quad (6.29)$$

where $\mathbf{P}(t)$ is a state probability vector for phase 1.

The system is ready to start the second phase only if at time t_1 it is in one of the states s_1, s_3, s_5. These events have probabilities $P_1(t_1)$, $P_3(t_1)$, $P_5(t_1)$ respectively. Then the probability that the system has success in the first phase and it is able to start second one is:

$$R(t_1) = P_1(t_1) + P_3(t_1) + P_5(t_1) \quad (6.30)$$

These probabilities can be added because the states s_1, s_3, s_5 are mutually exclusive. [38]

b) *Second phase: from t_1 to t_2.* The system is successful during the second phase if it starts in one of the states s_1, s_3, s_5 and evolves only through these states. In order to consider the whole mission as a success, the system at t_2 has to occupy state s_1, the only state suitable for both the second and the third phase.

The transition rate matrix for phase 2 can be obtained as before using Table 6.3:

$$\Lambda_2 = \begin{bmatrix} -\Sigma_1 & 0 & \mu_B & 0 & \mu_c & 0 & 0 & 0 \\ \lambda_A & -\Sigma_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_B & 0 & -\Sigma_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_B & \lambda_A & -\Sigma_4 & 0 & 0 & 0 & 0 \\ \lambda_C & 0 & 0 & 0 & -\Sigma_5 & 0 & 0 & 0 \\ 0 & \lambda_C & 0 & 0 & \lambda_A & -\Sigma_6 & 0 & 0 \\ 0 & 0 & \lambda_C & 0 & \lambda_B & 0 & -\Sigma_7 & 0 \\ 0 & 0 & 0 & \lambda_C & 0 & \lambda_B & \lambda_A & 0 \end{bmatrix} \tag{6.31}$$

The equation for the above evolution is

$$\dot{\mathbf{q}}(t) = \Lambda_2 \mathbf{q}(t) \tag{6.32}$$

where $\mathbf{q}(t)$ is a state probability vector for phase 2.

In order for the second phase to follow the first phase, the initial condition has to take into account the first phase. Therefore,

$$\begin{aligned} & P\{\text{1st phase success AND 2nd phase success}\} \\ = \; & P\{\text{2nd phase success} \mid \text{1st phase success}\} P\{\text{1st phase success}\} \\ = \; & \sum_{s_i \in s_{1,2}} P\{\text{2nd phase success} \mid \text{system in } s_i \text{ at } t_1\} \times P\{\text{ system in } s_i \text{ at } t_1\} \end{aligned}$$

where $s_{i,j}$ is a set of success states for both phases i and j.

Thus, the initial condition for (6.32) is:

$$\mathbf{q}(0) = [P_1(t_1) \; 0 \; P_3(t_1) \; 0 \; P_5(t_5) \; 0 \; 0 \; 0]^T \tag{6.33}$$

The system is now able to start the third phase, having successfully completed the first two phases with a probability:

$$R(t_2) = q_1(t_2 - t_1) \tag{6.34}$$

c) *Third phase: from t_2 to t_3.* In this phase the problem to solve is

$$\dot{\mathbf{Z}}(t) = \Lambda_3 \mathbf{Z}(t) \tag{6.35}$$

where $\mathbf{Z}(t)$ is a state probability vector for phase 3.

with initial condition

$$\mathbf{Z}(0) = [q_1(t_2 - t_1)\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T \tag{6.36}$$

$$\Lambda_3 = \begin{bmatrix} -\Sigma_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_A & -\Sigma_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_B & 0 & -\Sigma_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_B & \lambda_A & -\Sigma_4 & 0 & 0 & 0 & 0 \\ \lambda_C & 0 & 0 & 0 & -\Sigma_5 & 0 & 0 & 0 \\ 0 & \lambda_C & 0 & 0 & \lambda_A & -\Sigma_6 & 0 & 0 \\ 0 & 0 & \lambda_C & 0 & \lambda_B & 0 & -\Sigma_7 & 0 \\ 0 & 0 & 0 & \lambda_C & 0 & \lambda_B & \lambda_A & 0 \end{bmatrix} \tag{6.37}$$

The probability of success of the phased mission system is

$$P\{\text{success of the phased mission system}\} = \mathbf{Z}_1(t_3 - t_2). \tag{6.38}$$

6.3.2.1.2 Random mission phase change time

Many systems such as real-time control for aircraft and space vehicles in which the computing system is required to execute different sets of computational tasks during different phases of a control process, the duration of a phase is more realistically modelled by a random variable. Thus the modelling of a particular phase can be tailored not only to the computational demands of each phase but also to the relevant properties of the system that influence performance during that phase.

Alam and Al-Saggaf [38] consider that the phase-change times can be random. They propose two approaches to determine an appropriate description of the marginal distributions of the mission phase change times.

The first approach investigates a general formula for the joint probability density function of the mission phase change times which may be statistically dependent. The second approach models the mission phase change times as order statistics of a continuous random variable. The solution for probabilistic mission phase change times is similar to the case of deterministic approach except for initial conditions, state [38].

6.3.2.2 A non-homogeneous Markov model

Smotherman [39, 40] note that in the approach proposed by [38] the individual phase models are limited in coupling by the use of expected values for the components of the transformed probability vectors. Such models continue to make the assumptions that phase changes are state-independent and instantaneous. It is noted [39, 40] that any time-homogeneous Markov model is limited to the implicit assumption that state holding times, e.g. component failure times and repair times are exponentially distributed.

The modelling of a phased mission system by a single nonhomogeneous Markov model removes the major limitations of the traditional phased mission approach and greatly increases modelling flexibility and scope of practical application [40]. The model proposed by [40] provides for non-exponential component failure behaviour.

If $\{X(t)|t \geq 0\}$ is a finite state stochastic process with state probabilities $p_i(t) = P[X(t) = i]$, then by use of a Markov model the following differential equations can be derived [40]

$$p_i'(t) = \sum_j p_j(t)a_{ij}(t) \tag{6.39}$$

where $a_{ij}(t)$ is the transition rate from state i into state j, $a_{ii}(t)$ is the negative row sum of row i, $-\sum_{j=0,j\neq i}^{n-1} a_{ij}(t)$. The system of equations can be rewritten as

$$P'(t) = P(t)A(t) \tag{6.40}$$

where $P(t) = (p_0(t), p_1(t), \ldots, p_{n-1}(t))$ is the row vector of state probabilities and $A(T) = [a_{ij}(t)]_{n\times n}$ is the transition rate matrix.

The time-homogeneous Markov model is the special case in which all transition rates are independent of time, i.e. $A(t) = A$. The approach to phased mission analysis is based on two important modifications of the nonhomogeneous Markov model [39]:

1. The concept of a state transition is generalized to include phase changes, as well as failures and repairs.

2. Reward measures are incorporated into the model to provide more information for system effectiveness evaluation.

In this approach, different phases are represented as different subsets of states in the single model, and phase changes are represented by time-varying transitions among these subsets. Because of the single model framework, phase change transitions out of the different states in a given phase subset can have different rates or impulse functions. Thus, phase changes are state dependent.

The system of differential equations was solved by adapting the fifth order Runge-Kutta method, see [40]. The solver was extended to handle fixed-time phase changes and efficiently recompute the time-dependent transition rates needed by the ordinary differential equations (ODE) solver when performing derivative evaluations.

The major change to the Runge-Kutta method was the use of an event queue driver [40]. Phase changes are inserted as events and include information on the type of change, the existing state, the entry state(s), and the branching probabilities for multiple-entry states. A step-size control adjusts the next step in the solution so as to not overstep the next event. Multiple events are allowed at the same time.

Fixed-time phase changes do not affect the transition-rate matrix but cause an instantaneous transfer of probability from the existing state into the entry state(s). Phase change times having uniform distributions have beginning and ending events. The beginning event inserts an entry into a recalculation list of time-dependent transition rates, which is processed upon each derivative evaluation. Ending events remove the corresponding entry.

To bound the local error of each step in the solution, the adaptive step-size control of Runge-Kutta method was used. This is in addition to the event step size control mentioned above. Minimum and maximum step sizes are specified, and a fixed time phase change transition is performed near the ending time of a uniform distribution if the value of the associated transition rate grows too large. This is an instantaneous transfer of the residual probability of the existing state into the entry state(s) and it is used by the adaptive step size control whenever the step size required to meet the local error tolerance is smaller than a minimum specified step size. Using the error tolerance and minimum step size parameters, the accuracy of the solution can be increased at the expense of efficiency.

Repairs are not generally modelled by nonhomogeneous Markov systems [39]. This restriction is necessary since a repair is assumed to return the failure process of a component to time $t = 0$. It is noted [39] that for time-homogeneous Markov models this assumption does not present a difficulty since each transaction erases

The model proposed by [41] considers the problem in terms of the construction of the continuous-time discrete-state Markov model and uses a standard Markov-chain solution technique that is adapted to phased missions. The resulting state space is the union of the states in each independent phase, rather than sum. The technique combines all the phases into one model, and uses a fault tree to specify the reliability model of the system. The resulting Markov model can be used to calculate measures such as the probability of successfully completing a mission, the time dependent probability of failure, or the mean time between failures.

The approach is especially useful where several phases are repeated many times because each phase needs to be described only once. This approach applies where the transition rates (failure and repair rates) are constant, and where the phase change times are deterministic. If any of these criteria are not met and if the system is not very large, then the approach proposed by Smotherman (see 6.3.2.2) is appropriate.

To explain the method, the example depicted in Figure 6.8 is considered. The corresponding fault trees are shown in Figure 6.9. If each of fault trees were converted to Markov chains separately, the chains shown in Figure 6.10 would result. In Figure 6.10 for phase 1 all possible states for the system are shown. From state 111 (all components are working) system can go to states 011, 101 or 110 (one of the components is failed), but they are all successive states. Only one component can fail at a time. To reach the failure state (F), all components must fail. To reach the failure state components may fail in different order ($111 \rightarrow 011 \rightarrow 001 \rightarrow F$, $111 \rightarrow 101 \rightarrow 001 \rightarrow F$, etc.). In phase 2 if component A fails, the system fails ($111 \rightarrow F$). In phase 3 failure of any component will lead to system failure ($111 \rightarrow F$).

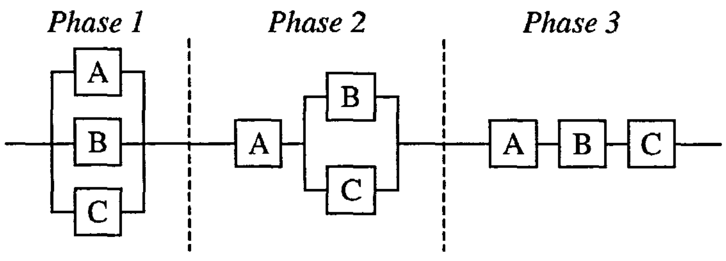


Figure 6.8: Example system

To combine all models into one, a multiplicative factor can be appended to each transition that will label it by the phase to which it belongs. That is, the transitions

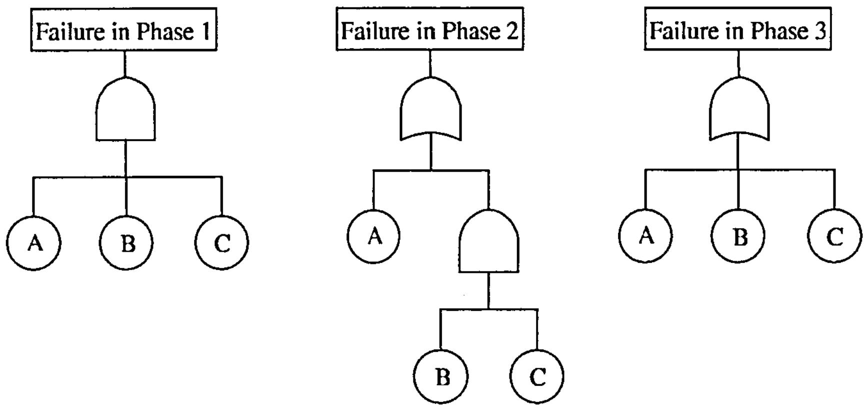


Figure 6.9: Fault trees for the example system

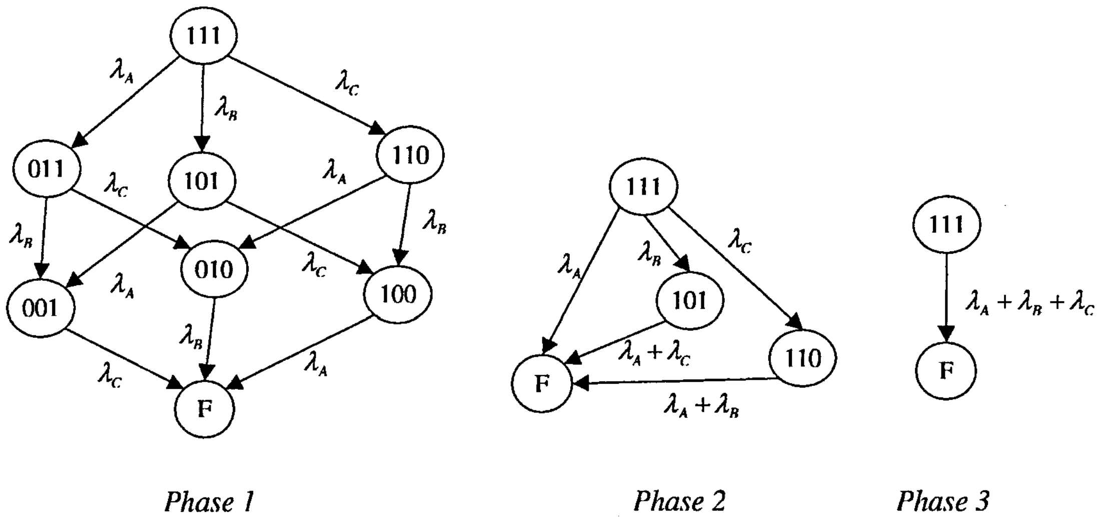


Figure 6.10: Markov model for each phase of the example system

in the model for phase i, every transition is multiplied by F_i. A combined Markov chain is formed whose state space is the union of the state spaces of the models of the separate phases, and whose transitions are the sum of the corresponding transitions of the models of the separate phases. The combined model for the example system is shown in Figure 6.11 where F_i labels the transitions that pertain to phase i.

Once generated the model can be solved using a standard numerical technique, with the following change. For the solution times that belong to phase i ($T_{i-1} \leq t \leq T_i$), F_i is set to one, and all other F_j, $j \neq i$, are set to zero. This assignment filters out any transition that does not belong to the current phase. Using this method, the state space does not change, and so the state probabilities need not undergo any

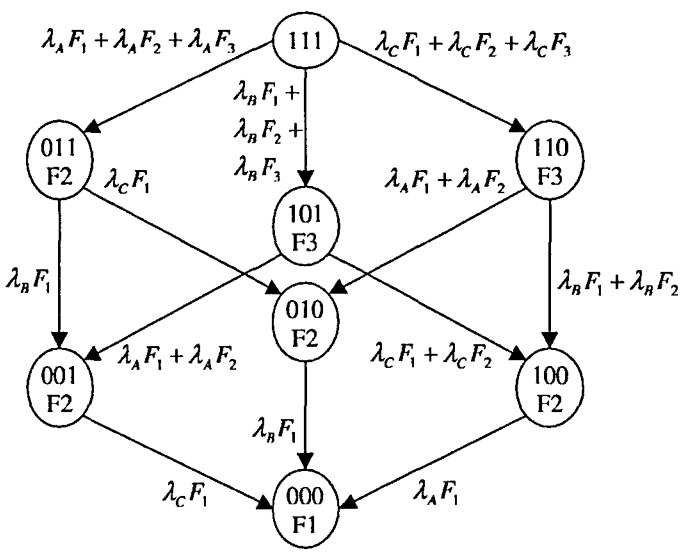

111
011 F2
110 F3
101 F3
010 F2
001 F2
100 F2
000 F1
$\lambda_A F_1 + \lambda_A F_2 + \lambda_A F_3$
$\lambda_C F_1 + \lambda_C F_2 + \lambda_C F_3$
$\lambda_B F_1 + \lambda_B F_2 + \lambda_B F_3$
$\lambda_C F_1$
$\lambda_A F_1 + \lambda_A F_2$
$\lambda_B F_1$
$\lambda_B F_1 + \lambda_B F_2$
$\lambda_A F_1 + \lambda_A F_2$
$\lambda_C F_1 + \lambda_C F_2$
$\lambda_B F_1$
$\lambda_C F_1$
$\lambda_A F_1$

Figure 6.11: Combined Markov model for all phases of the example system

transformation, but rather the transitions themselves are changing with the phase changes. The resulting model is still Markov, but it is no longer homogeneous since the transition values depend on global time. However, most standard numerical techniques apply to non-homogeneous Markov models.

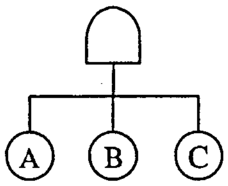
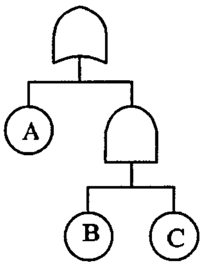
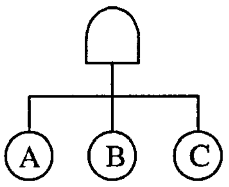
In general, separate models are not generated for each phase; rather the combined model is generated from the start.

6.3.2.4 Fault tree approach

A repairable multiphase system cannot be reduced to an equivalent single phase system, states Clarotti *et al* [37]. This is due to the fact that logical operations used for nonrepairable systems cannot be carried out if repair is foreseen. Considering the mission with the profile described in Table 6.4, the solution of the problem of evaluating the unreliability of the system is straight forward using the Monte Carlo simulation technique, see [37].

When using asynchronous simulation technique, the fault tree of the system whose reliability has to be calculated, is checked every time a failure occurs. Let t_i be the phase change time between phases i and $(i + 1)$. Suppose that when

Table 6.4: Mission profile

Phase No.	Time interval	System configuration (Block diagram)	System task	Parameter of interest	Maintenance policy
1	$(0,t_1)$		x	Reliability	Multiple
2	(t_1,t_2)		y	Reliability	Multiple
3	(t_2,t_3)		z	Reliability	Multiple

generating the failure times t_A, t_B and t_C the following situation occurs:

$$t_A < t_1$$
$$t_2 < t_B < t_3$$
$$t_C > t_3$$

The fault tree of the system in the first phase is checked and the system is found to be functioning. The repair time τ_A of component A is generated and suppose that is it $t_1 < t_A + \tau_A < t_B$. This situation corresponds to a system failure at t_1, due to A being in a failed state at the phase change. A normal Monte Carlo code would ignore this failure as the system is checked only when failure occurs. In order to avoid this situation, a check of system state is performed at the beginning of each phase, in order to verify if a system failure occurs at that time due to the change in configuration.[37]

In general, fault tree techniques can be used to find an analytical upper bound to system reliability, [37]. For highly reliable systems, for example systems of interest in the nuclear field, the expected number of failures is a close approximation for the unreliability [37].

Assumptions made by Clarotti *et al* are:

- a multiple repair policy (one repairman per item) is assumed in any case; this fact does not appreciably affect the result due to the large difference between times to failure and to repair;

- the upper bound is found by stopping the expected number of failures expansion to the first order terms.

Let $x(i,j)$ indicate the event that minimal cut set i, $1 \leq i \leq n$, occurs for the first time in phase j, $1 \leq j \leq n$. $x(i,j)$ is a null (impossible) event, if the i-th minimal cut set does not appear in the j-th phase.

The system unreliability at the mission time T is given by:

$$\bar{R}_s(T) = p\left\{\bigcup_{i=1}^{n}\left[\bigcup_{j=1}^{k} x(i,j)\right]\right\} \tag{6.42}$$

Then, according to the above assumption:

$$\bar{R}_s(T) \leq \sum_{i=1}^{n} p\left\{\bigcup_{j=1}^{k} x(i,j)\right\} \tag{6.43}$$

As the $x(i,j)$ are mutually exclusive with respect to j, for any i:

$$\begin{aligned} p\left\{\bigcup_{j=1}^{k} x(i,j)\right\} &= p\left\{x(i,j_1)\right\} + p\left\{x(i,j_2)|\bar{x}(i,j_1)\right\} p\left\{\bar{x}(i,j_1)\right\} \\ &\quad + p\left\{x(i,j_3)|\bar{x}(i,j_1)\bar{x}(i,j_2)\right\} p\left\{\bar{x}(i,j_1)\bar{x}(i,j_2)\right\} \\ &\quad + \ldots \end{aligned} \tag{6.44}$$

where

$p\left\{x(i,j_r)\right\}$ is the probability that the i-th minimal cut set first occurs in phase j_r;

j_r is the r-th phase in which the considered minimal cut set appears;

$p\{x(i,j_r)|\bar{x}(i,j_1),\ldots,\bar{x}(i,j_{r-1})\}$ is the conditional probability of the i-th minimal cut set that first occurs in its r-th possible phase, given that it did not occur in the previous $r-1$ possible phases;

$p\{\bar{x}(i,j_1),\ldots,\bar{x}(i,j_{r-1})\}$ is the probability that the i-th minimal cut set never occurred in its first $r-1$ phases.

When evaluating the terms of equation (6.44), the following three cases are considered:

a) none of the components of the i-th minimal cut set appears in the phases in which that minimal cut set does not appear, i.e. those components work only in the phases in which all of them give rise to the i-th minimal cut set. In this case it is possible that the i-th minimal cut set occurs at a phase change time due to a change in configuration. Then equation (6.44) reduces to:

$$\begin{aligned} p\left\{\bigcup_{j=1}^{k} x(i,j)\right\} &= \bar{R}_{x(i,j_1)} + \bar{R}_{x(i,j_2)}R_{x(i,j_1)} \\ &+\bar{R}_{x(i,j_3)}R_{x(i,j_2)}R_{x(i,j_1)} + \ldots \end{aligned} \tag{6.45}$$

b) all components of the i-th minimal cut set appear, i.e. working, in some other minimal cut set in phases in which i-th minimal cut set does not appear. In this case let the j_r-th phase be the first one in which the minimal cut set appears again after a certain number of phases in which it did not appear, but its components all worked belonging to other minimal cut set. In this case the term $p\{x(i,j_r)\bar{x}(i,j_{r-1}),\ldots,\bar{x}(i,j_1)\}$ may be split in to the sum of two mutually exclusive events, such as:

$$\begin{aligned} p\{x(i,j_r,)\bar{x}(i,j_{r-1}),\ldots,\bar{x}(i,j_1)\} &= p\{x_b(i,j_r),\ldots,\bar{x}(i,j_1)\} \\ &+p\{x_d(i,j_r),\bar{x}_b(i,j_r),\bar{x}(i,j_{r-1}),\ldots,\bar{x}(i,j_1)\} \end{aligned} \tag{6.46}$$

where the subscripts b and d mean respectively at the beginning and during the possible j_r-th phase of i-th minimal cut set. The cut set may occur at the beginning of the phase due to the change of configuration.

$$\begin{aligned} &p\{x_b(i,j_r),\bar{x}(i,j_{r-1}),\ldots,\bar{x}(i,j_1)\} = \\ &= p\{x_b(i,j_r)|\bar{x}(i,j_{r-1}),\ldots,\bar{x}(i,j_1)\}\cdot p\{\bar{x}(i,j_{r-1}),\ldots,\bar{x}(i,j_1)\} \end{aligned} \tag{6.47}$$

The conditional probability in the above expression is upper bounded by the unavailability of the i-th minimal cut set at the beginning of the possible j_r-th phase, taking into account the previous evolution of the components.

Furthermore,

$$p\left\{\bar{x}(i,j_{r-1}),\ldots,\bar{x}(i,j_1)\right\} = \prod_{h=j_1}^{j_{r-1}} R_{x(ij)} \tag{6.48}$$

and

$$\begin{aligned}
&p\left\{x_d(i,j_r),\bar{x}_b(i,j_r),\bar{x}(i,j_{r-1}),\ldots,\bar{x}(i,j_1)\right\} = \\
&= p\left\{x_d(i,j_r)|\bar{x}_b(i,j_r),\bar{x}(i,j_{r-1}),\ldots,\bar{x}(i,j_1)\right\}\cdot \\
&\cdot p\left\{\bar{x}_b(i,j_r)|\bar{x}(i,j_{r-1}),\ldots,\bar{x}(i,j_1)\right\}\cdot \\
&\cdot p\left\{\bar{x}(i,j_{r-1}),\ldots,\bar{x}(i,j_1)\right\}
\end{aligned} \tag{6.49}$$

The first factor in equation (6.49) is the unreliability of the i-th minimal cut set during its j_r-th phase. The second factor is upper bounded by 1. The third factor is given by equation (6.48).

c) some of the components of the i-th minimal cut set appear in some other minimal cut sets in the phases in which i-th minimal cut set does not appear. It is the intermediate case. If, among the components which do not work in phases between j_{r-1}-th and the j_r-th, at least one has a mean time to repair much shorter than the time interval between the end of the former and the beginning of the latter, the probability of failure at the j_r-th phase change time is zero.

Case c) can be handled like case a), see [37].

With the hypothesis of exponentially distributed time to failure and time to repair, the unreliability $\bar{R}_{x(i,j)}$ is:

$$\bar{R}_{x(i,j)} \leq \sum_{h=1}^{c} \lambda_h \int_{t_{j-1}}^{t_j} [1 - q_h(t)] \prod_{s=h}^{c} q_s(t)dt$$

where h and s indicate generic component of the i-th minimal cut set and $q_r(t)$ is the unavailability of the r-th component calculated taking into account the fact that, if the component worked previously in some other phase, its initial unavailability q_s is different from zero.

6.4 Summary

According to Burdick *et al* one of the most important problems in systems unreliability analysis is the phased mission problem. Most reliability techniques consider only systems undergoing a single phase, but multiple phases are very common in many systems. Phased mission analysis is recognised as the appropriate reliability analysis method for large number of problems, including systems in nuclear, aerospace, chemical, electronical and other industries. Analysis of a phased mission system encounters difficulties which are not present with a single-phase systems as system configuration, phase duration, failure rates of components may vary from phase to phase.

Esary and Ziehms [2] consider a multiphase system with nonrepairable components. They transform the original problem to a single phase mission using basic event transformation and cut set cancellation. Using this method the reliability of the system during each phase cannot be calculated as the components may be removed from earlier phases if they are met in minimal cut sets of later phases. Pedar and Sarma [31] extended the method for multiobjective phased mission systems. The method proposed by Esary and Ziehms was also used by Xing and Dugan [36] for a generalized phased mission system analysis methodology. They were considering phased mission systems which have combinatorial phase requirements and imperfect coverage and integrated several techniques (BDD-based algorithm for system reliability analysis, basic event transformation).

A different method was proposed by Dazhi and Xiaozhong [34]. They don't use basic event transformation, but introduce generalized intersection and union concept and add them to the list of Boolean algebra principles to consider the time-dependent effect between cut sets in different phases.

A phased mission reliability technique based on Boolean algebraic methods was proposed by Somani and Trivedi [29]. One phase is handled at a time and then the unreliability of the entire mission is calculated. Phase algebra is developed to account for the effects of variable configurations and success criteria from phase to phase. According to Somani and Trivedi, the technique can be very useful for a large class of the systems where the system behaviour can be described using fault trees. Later the method was extended by Ma and Trivedi [28] introducing an algorithm based on the sum of disjoint phase products to analyze the system unreliability.

For repairable systems two methods are considered: fault tree methods and Markov methods. Clarotti *et al* [37] note, that a fault tree approach gives an exact result only if a complete independence between systems components can be assumed. If the system is repairable, then only an upper bound can be found using the fault tree approach. According to Clarotti *et al* [37], the exact solution of the problem can be found using a Markov approach. The solution for the case when the phase change times are deterministic was proposed by Clarotti *et al* and for the stochastic phase change times method was proposed by Alam and Al-Saggaf [38]. Other approaches to Markov method were proposed by Smotherman [39, 40] and Dugan [41].

7. PHASED MISSION ANALYSIS USING THE CAUSE-CONSEQUENCE DIAGRAM METHOD

7.1 Cause-Consequence Analysis

The objective of a cause-consequence diagram is to evaluate the likelihood or frequency of each outcome that can result from the critical event. As discussed in Chapter 4, when the probabilities of the decision boxes of the cause-consequence diagram are independent the quantification can be done in the following steps:

1. Assign probabilities to each outlet branch stemming from the decision box (possibly by quantifying the relevant fault tree).

2. The probability of any sequence is obtained by multiplying the relevant probabilities associated with each decision box exit path in that sequence.

3. The probability of any consequence is obtained by summing the probabilities of each sequence terminating in that consequence.

However, this procedure cannot be used if the failures in each decision box are not independent. Dependencies can exist in cause-consequence diagrams due to repeated or inconsistent events and these must be dealt with before quantification.

In the case of repeated events the same failure event exists in more than one fault tree structure on the same path of a cause-consequence diagram. To deal with a common event failure the following algorithm is suggested, Andrews and Ridley [6, 23].

7.2 Phased Mission Analysis Using Cause-Consequence Diagram

To illustrate phased mission analysis using the cause-consequence diagram method, consider the example of the non-repairable system shown in Figure 7.1.

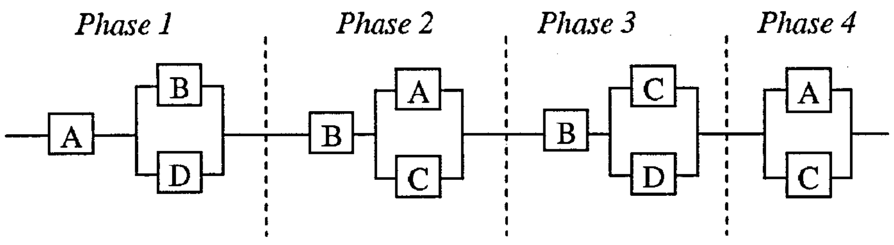


Figure 7.1: Simple phased mission system

In this example, the system is required to work successfully through all four phases to complete the mission. In phase 1 component A along with B or D are required to work. Failure of component A or components B and D would lead to mission failure. If phase 1 is completed successfully, the system enters phase 2, etc. To complete the mission successfully, the system requirements must be satisfied for each phase.

Considering the phases separately, the fault trees representing individual phase failures are shown in Figure 7.2.

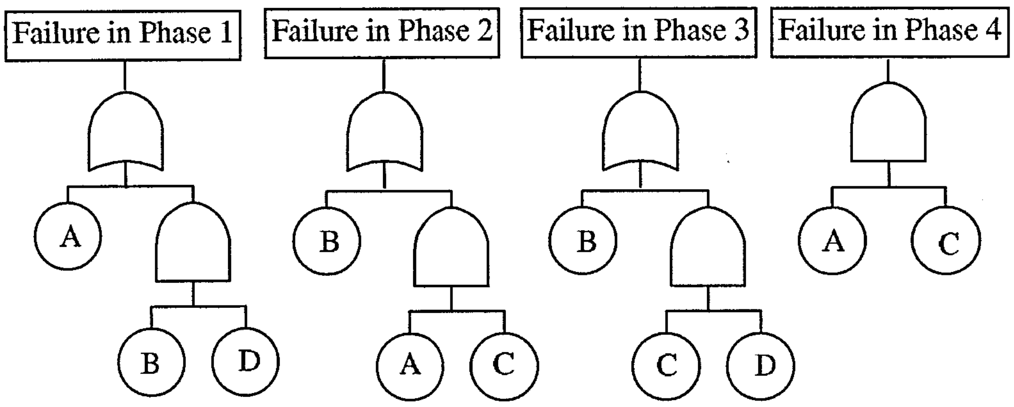


Figure 7.2: Fault tree representation of individual phase failures

If a cause-consequence analysis is to be performed the challenge is to develop the diagram in an efficient way. The entry point of the diagram is the start of the mission. Consequence events are the failure during each of the phases or mission success.

7.3 Cause-Consequence Diagram Construction Methods

Two methods have been developed in order to generate the cause-consequence diagram efficiently and analysis has been performed. The advantages and disadvantages of each method are discussed below.

7.3.1 Method 1

One way of constructing the cause-consequence diagram is to assume a certain order in which the component failure events will be considered. For this example assume the following order:

$$A1 < B1 < D1 < C1 < B2 < A2 < C2 < D2 < B3 < C3 < D3 < A3 < A4 < C4,$$

where A1 means component A fails in phase 1, etc., A2 - component A fails in phase 2 having worked successfully through phase 1, etc.

Following the order of component failures given, each component failure event is added to the diagram as a decision box. Therefore, considering component A in phase 1: if component A fails, then from the fault tree for phase 1 it can be seen that the system fails in that phase, hence the mission is failed. The CCD is constructed by inserting a decision box asking ‘A fails in phase 1’. The YES branch represents system failure and therefore a consequence is added. Following the NO branch as component A is functioning throughout phase 1, from the ordering component B in phase 1 is considered. If component B fails, then component D in phase 1 is considered. If D fails in phase 1, then mission is failed in phase 1. If component D does not fail in phase 1, the mission progresses to phase 2, however since B is failed in phase 1 the system fails immediately on entering phase 2, see fault tree for phase 2 in Figure 7.2. If component B works throughout phase 1, component D is considered. The successful functioning of component D throughout phase 1 following the success of components A and B means the mission progresses to phase 2. Components are again considered one after the other until the conditions are met for phase 2 failure or progression to phase 3. The complete cause-consequence diagram is shown in Figures 7.3 - 7.4. In the diagram, consequences F1, F2, F3 and F4 mean mission failure due to system failure in phase 1, 2, 3 and 4 respectively, C represents mission success.

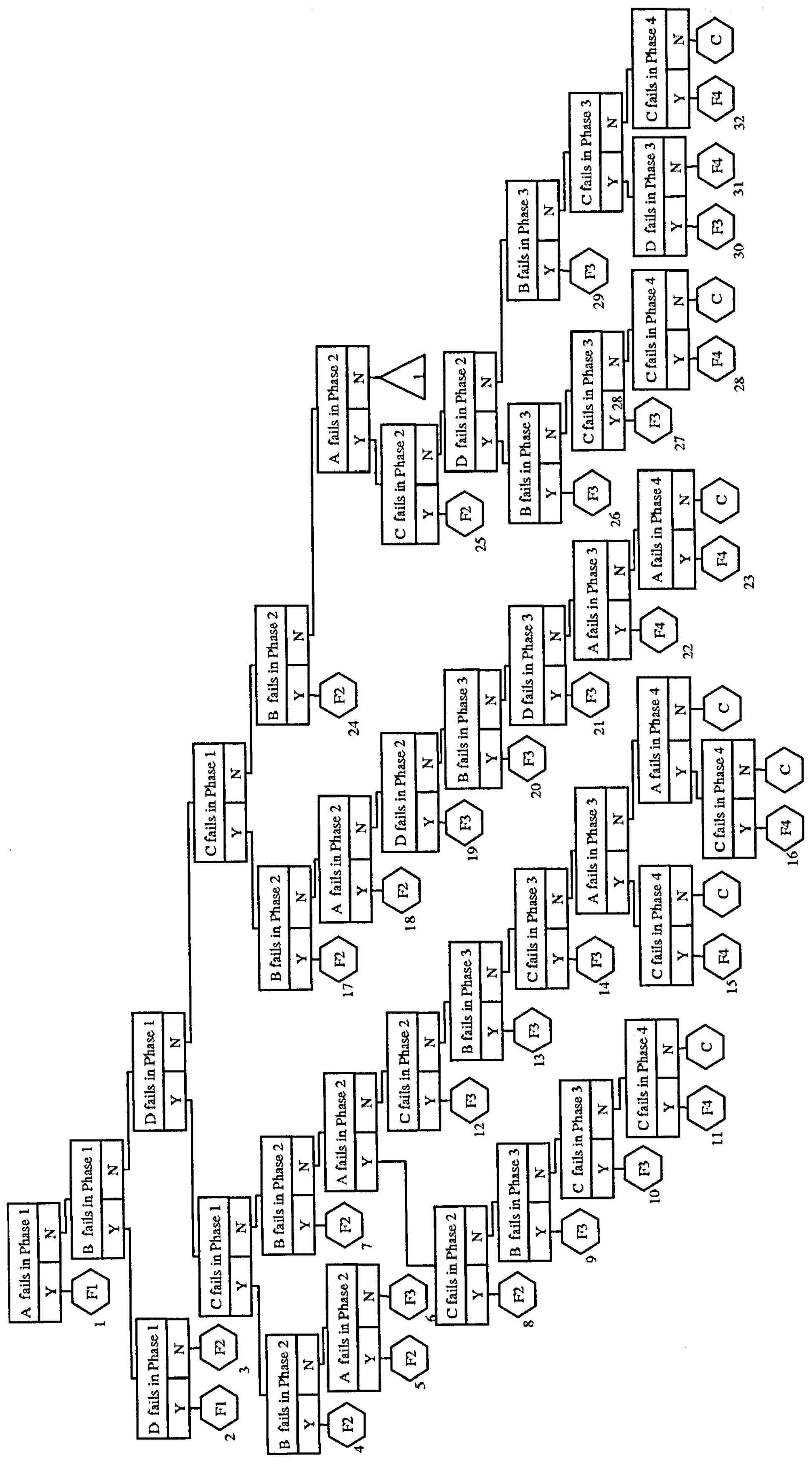


Figure 7.3: Cause-consequence diagram for system shown in Figure 7.1

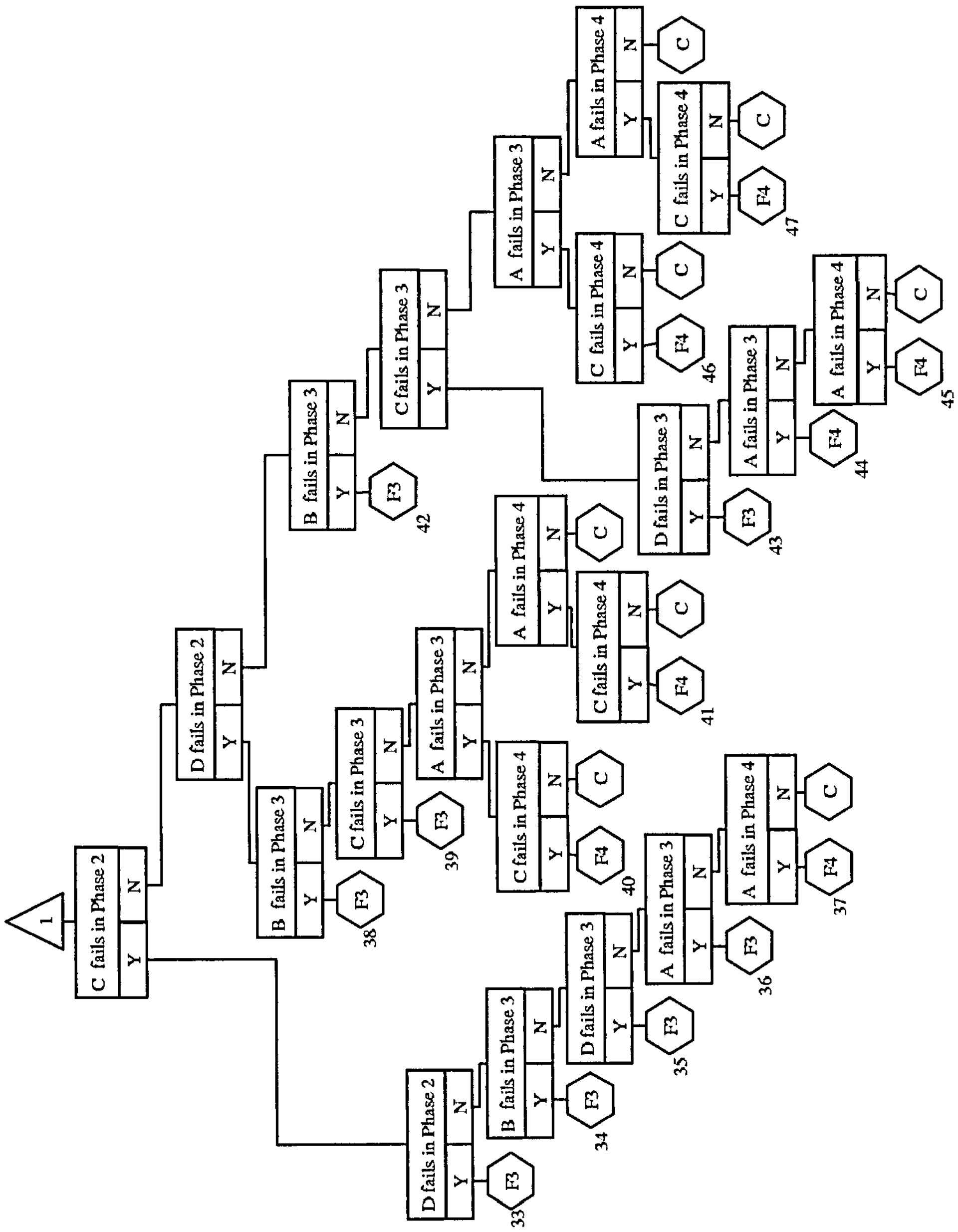


Figure 7.4: Continued: Cause-consequence diagram for system shown in Figure 7.1

7.3.2 Qualitative Analysis

The qualitative analysis of the cause-consequence diagram will produce the list of causes for each outcome condition. Conditions causing any outcome event are established by investigating each decision box on the path to the outcome and listing the component failure or success in the phase as indicated by the exit path from the decision box. In the example considered here there are 47 failure outcomes, numbered 1 - 47 in Figure 7.3 - 7.4. The component conditions for each of these outcomes was determined, as an example the failure outcomes resulting in mission failure in phases 1 and 2 are listed below.

F1 - Mission failure in Phase 1

1. $A1$
2. $\overline{A1} \wedge B1 \wedge D1$

F2 - Mission failure in Phase 2

3. $\overline{A1} \wedge B1 \wedge \overline{D1}$
4. $\overline{A1} \wedge \overline{B1} \wedge D1 \wedge C1 \wedge B2$
5. $\overline{A1} \wedge \overline{B1} \wedge D1 \wedge C1 \wedge \overline{B2} \wedge A2$
7. $\overline{A1} \wedge \overline{B1} \wedge D1 \wedge \overline{C1} \wedge B2$
8. $\overline{A1} \wedge \overline{B1} \wedge D1 \wedge \overline{C1} \wedge \overline{B2} \wedge A2 \wedge C2$
17. $\overline{A1} \wedge \overline{B1} \wedge \overline{D1} \wedge C1 \wedge B2$
18. $\overline{A1} \wedge \overline{B1} \wedge \overline{D1} \wedge C1 \wedge \overline{B2} \wedge A2$
24. $\overline{A1} \wedge \overline{B1} \wedge \overline{D1} \wedge \overline{C1} \wedge B2$
25. $\overline{A1} \wedge \overline{B1} \wedge \overline{D1} \wedge \overline{C1} \wedge \overline{B2} \wedge A2 \wedge C2$

Each failure mode in the list contains a progression of states for the same component. For example, outcome 4 has component B working throughout phase 1, $\overline{B1}$, and then failing in phase 2, $B2$. The list can therefore be simplified. When there is a situation where a component is working throughout phase, but fails in a later phase, it is unnecessary to include the working state in the list as the failure in later phase implies that it was working before. This is the situation in several outcomes listed above. The simplified list of mission failures in phase 1 and 2 is listed below:

F1 - Mission failure in Phase 1

1. $A1$
2. $\overline{A1} \wedge B1 \wedge D1$

F2 - Mission failure in Phase 2

3. $\overline{A1} \wedge B1 \wedge \overline{D1}$
4. $\overline{A1} \wedge D1 \wedge C1 \wedge B2$
5. $\overline{B1} \wedge D1 \wedge C1 \wedge \overline{B2} \wedge A2$
7. $\overline{A1} \wedge D1 \wedge \overline{C1} \wedge B2$
8. $\overline{B1} \wedge D1 \wedge \overline{B2} \wedge A2 \wedge C2$
17. $\overline{A1} \wedge \overline{D1} \wedge C1 \wedge B2$
18. $\overline{B1} \wedge \overline{D1} \wedge C1 \wedge \overline{B2} \wedge A2$
24. $\overline{A1} \wedge \overline{D1} \wedge \overline{C1} \wedge B2$
25. $\overline{B1} \wedge \overline{D1} \wedge \overline{B2} \wedge A2 \wedge C2$

7.3.3 Quantitative Analysis

The reduced or simplified lists of component conditions leading to each outcome are in an appropriate form for quantification. This is because each of the outcome event sequences are mutually disjoint. Under these conditions the probability of achieving any particular phase failure outcome is the sum of the probabilities leading to that outcome.

Quantification of the diagram starts with the calculation of the probabilities for each event X failing in phase i having worked throughout the previous phases, $P(X_i)$. The probabilities are evaluated over the relevant phase period by integrating the component failure density function $f_X(t)$. Therefore if phase i is from t_{i-1} to t_i

$$P(X_i) = \int_{t_{i-1}}^{t_i} f_X(t)\,dt \tag{7.1}$$

When a component is working in two (or more) consecutive phases, for example 'component A works through phase 1' and 'component A works through phase 2', probability that 'A' works through both phases is calculated as:

$$P\left(\overline{A1} \wedge \overline{A2}\right) = 1 - P(A1) - P(A2)$$

Therefore, for the example considered, the probability of mission failure would be given by:

$$P(mission\ \ failure) = P(F1) + P(F2) + P(F3) + P(F4)$$

where

$$\begin{aligned}
P(F1) &= P(A1) + P\left(\overline{A1}\right) P(B1) P(D1) \\
P(F2) &= P\left(\overline{A1}\right) P(B1) P\left(\overline{D1}\right) + P\left(\overline{A1}\right) P(D1) P(C1) P(B2) + \\
&\quad + (1 - P(B1) - P(B2)) P(D1) P(C1) P(A2) + \\
&\quad + P\left(\overline{A1}\right) P(D1) P\left(\overline{C1}\right) P(B2) + \\
&\quad + (1 - P(B1) - P(B2)) P(D1) P(A2) P(C2) + \\
&\quad + P\left(\overline{A1}\right) P\left(\overline{D1}\right) P(C1) P(B2) + \\
&\quad + (1 - P(B1) - P(B2)) P\left(\overline{D1}\right) P(C1) P(A2) + \\
&\quad + P\left(\overline{A1}\right) P\left(\overline{D1}\right) P\left(\overline{C1}\right) P(B2) + \\
&\quad + (1 - P(B1) - P(B2)) P\left(\overline{D1}\right) P(A2) P(C2) = \\
&= P\left(\overline{A1}\right) P(B1) P\left(\overline{D1}\right) + P\left(\overline{A1}\right) P(B2) + \\
&\quad + (1 - P(B1) - P(B2)) P(C1) P(A2) + \\
&\quad + (1 - P(B1) - P(B2)) P(A2) P(C2)
\end{aligned}$$

Expressions were also determined for $P(F3)$ and $P(F3)$ and then the probability of mission failure determined.

The result obtained was found to be the same as that using the fault tree method and BDD analysis proposed by La Band and Andrews [3].

The method presented allows straight forward construction of the cause-consequence diagram. In the example presented the size of the problem was not obstructing to follow every branch through. With bigger systems it can get more difficult to construct the cause-consequence diagram manually and the aid of a computer may be needed.

7.3.4 Method 2

It can be seen that the quantification of the cause-consequence analysis diagram for phased missions is efficient and straight forward. The difficulty is in initially obtaining the cause-consequence diagram in the first place. For the simple system shown in Figure 7.1 it was straight forward, but as the size and complexity of the system increases so does the cause-consequence diagram. An alternative method is suggested below, which could be automated to ease the construction of larger cause-consequence diagrams.

The alternative method is based on the fact that the whole cause-consequence diagram contains events which are both repeated and inconsistent for each component.

The first step of the second method of cause-consequence diagram construction is shown in Figure 7.5 with corresponding fault trees shown in Figure 7.6. The diagram is at this stage at a high level of abstraction and just considers phase failure. The first decision box contains the failure event 'Phase 1 fails', if it is true, then mission fails in phase 1 (failure F1). If the system works successfully throughout phase 1, it progresses to phase 2. If the system fails in phase 2, then the cause-consequence diagram terminates with failure event 'Mission fails in phase 2' (F2). This is repeated for phases 3 and 4. If the system does not fail in phase 4, then the mission is completed successfully. Phase failure causes are developed using fault tree analysis and the relevant fault trees are attached to the YES branches of the decision box.

The fault trees in Figure 7.6 are obtained using the basic event transformation introduced by Esary and Ziehms [2] as described in Section 6.3.1.1.

As the cause-consequence diagram contains both repeated and inconsistent failure events, these must be extracted one by one following the normal cause-consequence analysis procedure described earlier. An order must be assumed in which the component failure events are to be considered for extraction. For this example a different order of component failure events is assumed than for Method 1:

$$A1 < B1 < B2 < A2 < C1 < C2 < B3 < C3 < D1 < D2 < D3 < A3 < A4 < C4$$

Then following the normal cause-consequence analysis process the event 'A fails in phase 1' (A1), as it is both a repeated and an inconsistent event, can be extracted

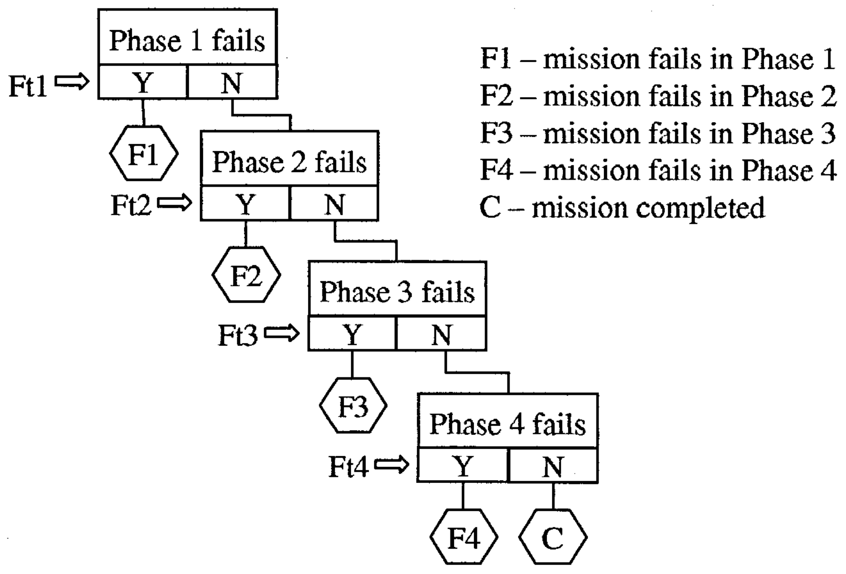

Phase 1 fails
Y
N
Ft1
F1
Phase 2 fails
Ft2
F2
Phase 3 fails
Ft3
F3
Phase 4 fails
Ft4
F4
C
F1 – mission fails in Phase 1
F2 – mission fails in Phase 2
F3 – mission fails in Phase 3
F4 – mission fails in Phase 4
C – mission completed

Figure 7.5: Construction of cause-consequence diagram - step 1

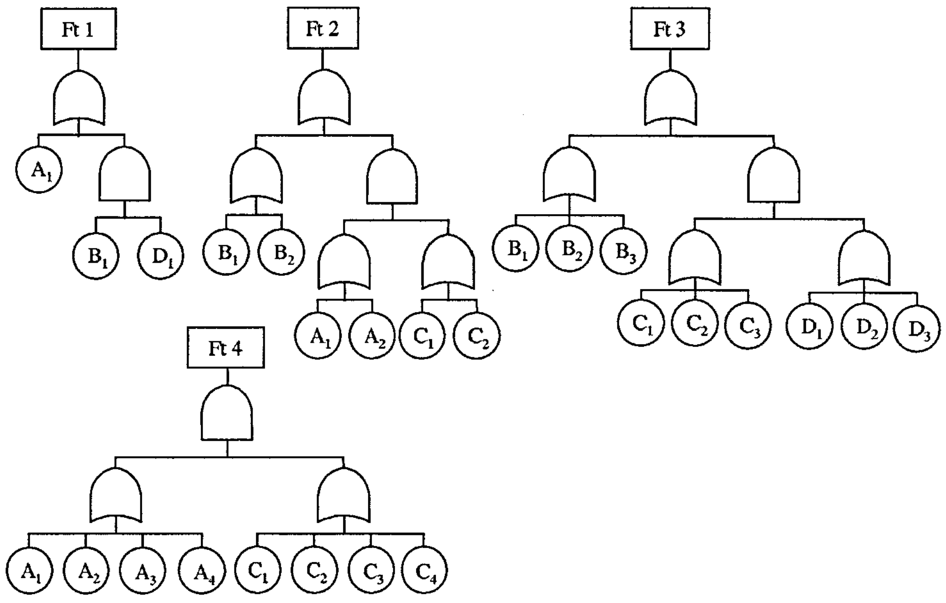


Figure 7.6: Fault trees for the cause-consequence diagram shown in Fig.7.5

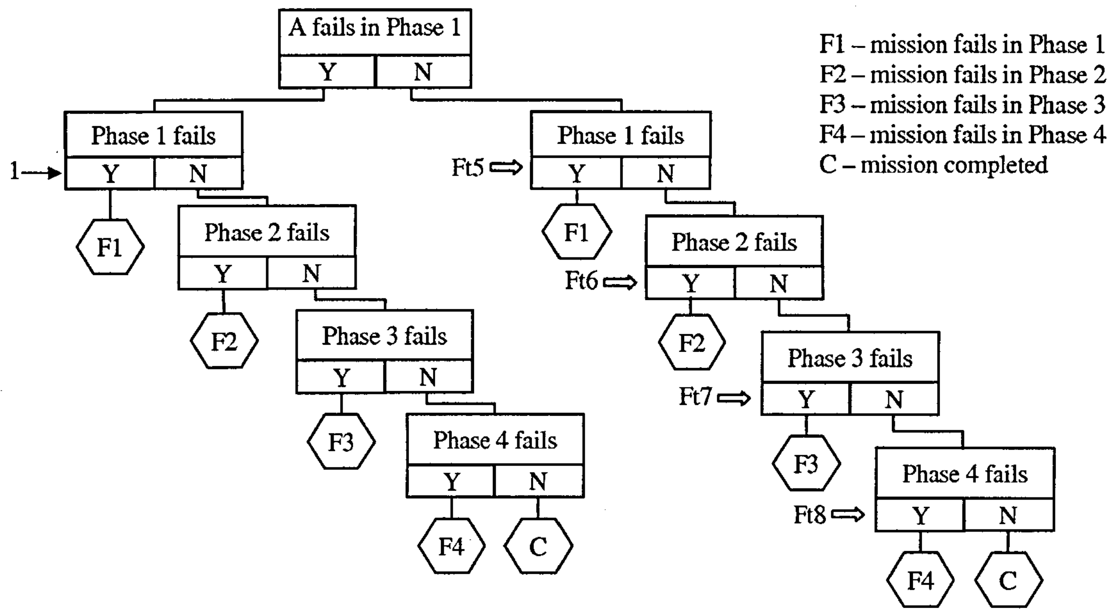


Figure 7.7: Construction of cause-consequence diagram - step 2

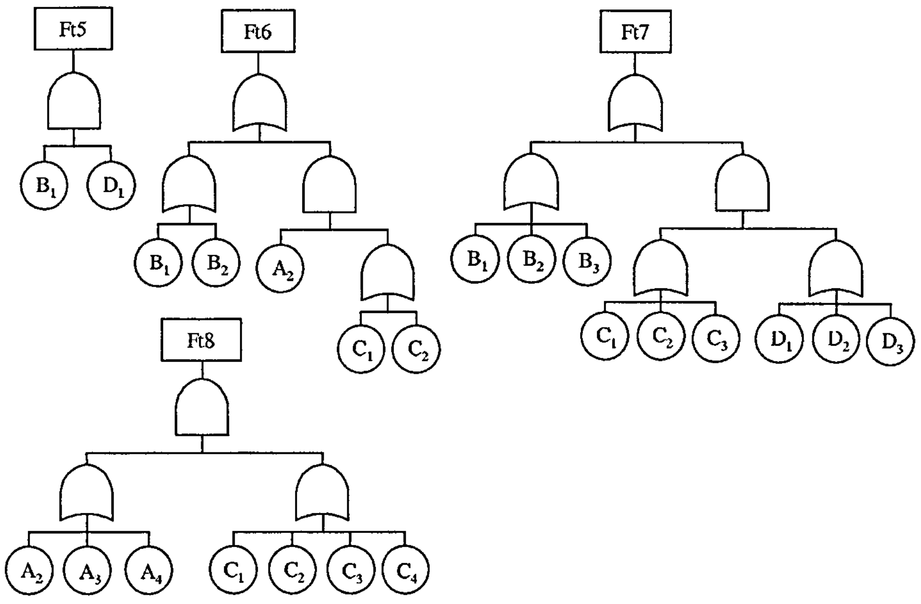


Figure 7.8: Fault trees for the cause-consequence diagram shown in Fig.7.7

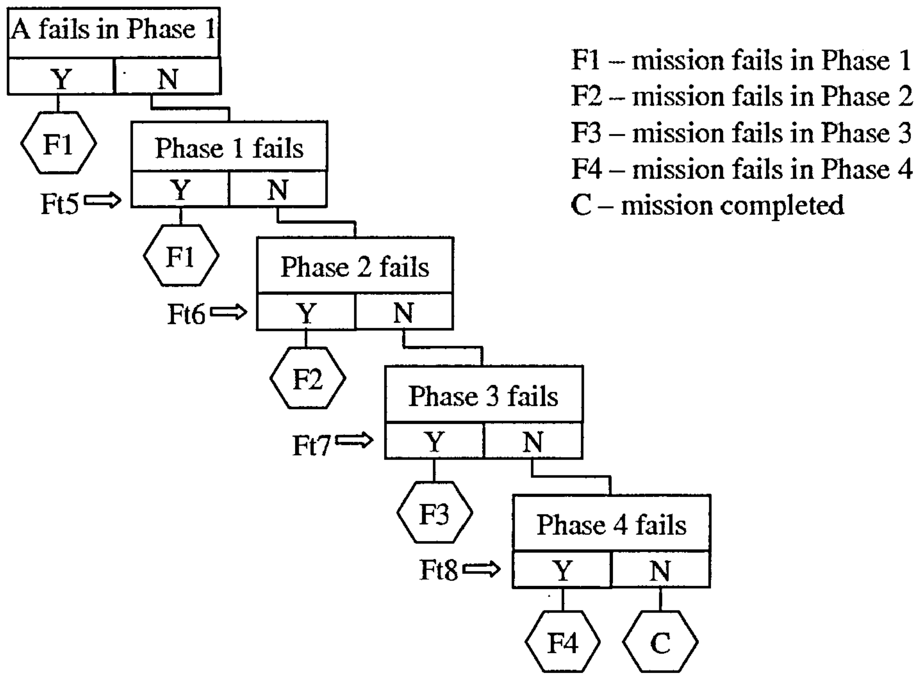


Figure 7.9: Construction of cause-consequence diagram - step 2 (reduced)

from fault tree structures and placed in a new decision box preceding the first decision box that contains it ('Phase 1 fails'). Then the diagram is duplicated on both outlet branches and following the YES branch all occurrences of the event are set to TRUE, and following the NO branch all occurrences of this event are set to be FALSE. The cause-consequence diagram for this step is shown in Figure 7.7 and the fault trees in Figure 7.8. As the probability of event 'Phase 1 fails' on the YES branch of the decision box 'A fails in phase 1' is 1, the diagram can be reduced (Figure 7.9).

Following the order of component failure events given the process is repeated and events are extracted from fault tree structures and placed in new decision boxes. The diagram is developed using normal cause-consequence analysis process. The final cause-consequence diagram obtained following this procedure is shown in Figure 7.10.

Once the cause-consequence diagram is obtained the same procedures for qualitative and quantitative analysis as for Method 1 should be followed. The cause-consequence diagram obtained using Method 2 gives the same result for the mission failure probability as the one obtained using Method 1.

From these two different methods it can be noticed that component ordering is important, as different variable orderings will produce different size diagrams. Both methods will give the same quantitative results, but they might produce different cause-consequence diagrams, as is also the case for binary decision diagrams. For example, using a different order of component failure events in Method 2 gives only 32 system outcome events compared with 47 in Method 1. The efficiency of the method will be very much influenced by the ordering chosen as this will influence the size of he cause-consequence diagram. The smaller the size of the diagram, the faster computing time would be to obtain the reliability/unreliability of the system.

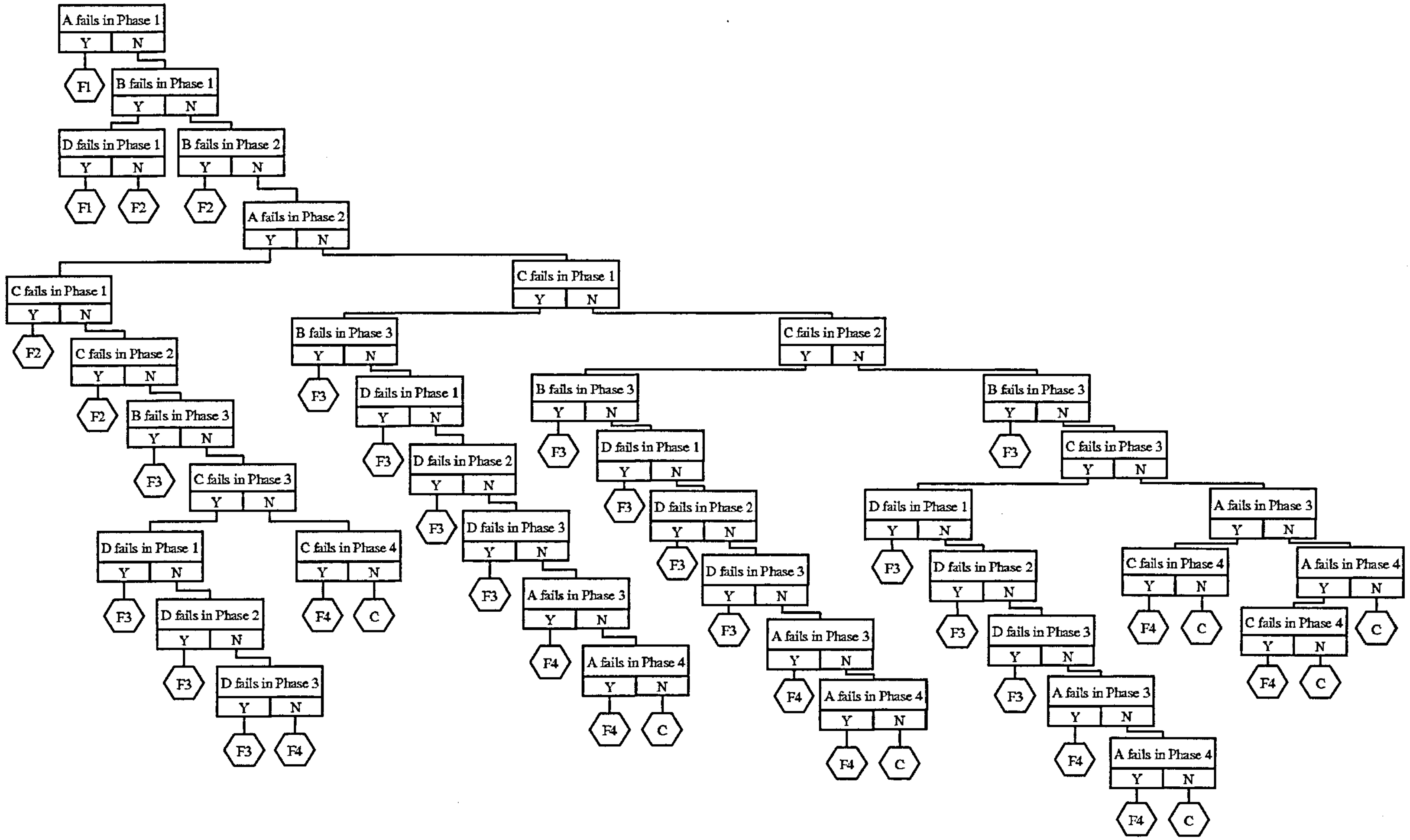


Figure 7.10: Final cause-consequence diagram

7.4 Program description

To assist with the cause-consequence diagram construction and analysis for phased mission systems a computer program was written. Method 1 was chosen for this code implementation.

The program consists of the following sections:

- Data input. The inputs are read from text files, which contain the following information about the phased mission:
  - number of phases;
  - list of text files with phase fault trees;
  - a text file with component failure probabilities.

  The fault tree text files are organised in the following way:
  - gate name;
  - gate type: 1 for 'And', 0 for 'Or';
  - number of basic events connected to the gate;
  - number of gates connected to the gate;
  - list of basic events;
  - list of gates.
- Basic event transformation. In this procedure basic events in all phases are replaced by an 'Or' combination of failure events for that and all preceding phases. For the phase 1 the new 'Or' gate would have only 1 event and therefore is simplified to omit the unnecessary gate.
- Cause-consequence diagram construction. To construct the cause-consequence diagram an order in which the basic events will be considered needs to be chosen. The options given are either to consider basic events in the same order as they are read from the input files or to enter a different ordering. Once the ordering has been chosen, the program places basic events in the decision boxes and depending if the top event of the phase fault tree occurs or not, they are attached to the relevant outcome of the previous decision box.

- Cause-consequence diagram minimisation. Once the cause-consequence diagram is constructed there could be redundant decision boxes where the phase failure would occur regardless whether the basic event fails or not. These boxes are removed.

- Cause-consequence diagram analysis. Once the final cause-consequence diagram is obtained, the probabilities are assigned to the decision boxes and the probabilities of outcomes are calculated. The probabilities are read from a text file which lists all basic events in the phase and the corresponding failure probabilities. The results are printed on screen and are also saved in a text file for ease of reference.

7.5 The use of ordering schemes in construction of CCD and BDD

Eight of the ordering schemes described in Chapter 5 were used to analyze the importance of variable ordering for construction of CCD and BDD. The schemes are:

- Modified top-down
- Modified depth-first
- Modified priority depth-first
- Depth-first, with number of leaves
- Non-dynamic top-down weights
- Dynamic top-down weights
- Bottom-up weights
- Event criticality

For cause-consequence diagrams the performances of the schemes were assessed in two ways:

- Component failures are ordered in each phase separately. In this case basic event transformation is applied to fault trees of each phase separately and then ordering schemes used to produce an order in which component failures should be considered. This enables the exact phase failure probability to be obtained as well as mission failure and mission success probabilities.

- Component failures are ordered for the whole mission. Basic event transformation is applied to fault trees at each phase as before, but before ordering the failures, all phases are joined in one fault tree under an 'OR' gate. This enables the mission failure and mission success probabilities to be calculated, but it does not give the probabilities of phase failure.

In each case the number of consequence boxes and total number of boxes (decision boxes + consequence boxes) was recorded.

For the construction of binary decision diagrams the failures were ordered separately for each phase.

The performance results of ordering schemes is shown in tables 7.1 - 7.3 below. As can be noticed from the tables below, structural importance measure, described in Section 2.6.1.1 was applied to component ordering for just a few examples at is quite computationally intensive. For the examples to which it was applied it did not produce the best results.

The examples used to compare the efficiency of the ordering schemes are as follow:

Example 1 Three phases, three components. Phase 1 - one gate, phase 2 - two gates, phase 3 - one gate

Example 2 Three phases, five components. Phase 1 - two gate, phase 2 - three gates, phase 3 - two gates

Example 3 Four phases, four components. Phase 1 - two gates, phase 2 - two gates, phase 3 - two gates, phase 4 - one gate

Example 4 Four phases, four components. Phase 1 - two gates, phase 2 - two gates, phase 3 - two gates, phase 4 - one gate. The difference between this example and the one above is that all gates have been inverted - gate 'AND' was changed to gate 'OR' and gate 'OR' to gate 'AND'

Example 5 Four phases, five components. Phase 1 - two gate, phase 2 - three gates, phase 3 - two gates, phase 4 - three gates

Example 6	Four phases, five components. Phase 1 - one gate, phase 2 - four gates, phase 3 - one gate, phase 4 - two gates
Example 7	Five phases, five components. Phase 1 - two gate, phase 2 - three gates, phase 3 - two gates, phase 4 - three gates, phase 5 - one gate
Example 8	Five phases, five components. Phase 1 - one gate, phase 2 - four gates, phase 3 - one gate, phase 4 - two gates, phase 5 - two gates
Example 9	Four phases, fourteen components. Phase 1 - one gate, phase 2 - five gates, phase 3 - five gates, phase 4 - six gates
Example 10	Five phases, fourteen components. Phase 1 - one gate, phase 2 - five gates, phase 3 - five gates, phase 4 - six gates, phase 5 - two gates
Example 11	Three phases, nine components. Phase 1 - three gate, phase 2 - six gates, phase 3 - three gates

	Modified Top-Down	Modified Depth-First	Modified Priority Depth-First	Depth-First, with Number of Leaves	Non-dynamic Top-Down Weights	Dynamic Top-Down Weights	Bottom-Up Weights	Event Criticality	Structural Importance
	1	2	3	4	5	6	7	8	9
Example 1	14	16	14	14	14	14	14	14	14
Example 2	99	150	99	150	99	99	150	99	111
Example 3	64	76	64	76	64	64	64	64	67
Example 4	119	151	119	151	119	119	151	119	
Example 5	472	506	472	506	472	472	506	472	484
Example 6	202	445	202	445	202	202	202	202	
Example 7	995	989	955	989	955	955	989	955	967
Example 8	575	889	575	889	575	575	575	575	
Example 9	456	1681	365	1681	456	456	461	456	
Example 10	669	2191	600	2191	669	669	669	669	
Example 11	3532	4862	4044	4862	3532	3562	4746	3532	

Table 7.1: CCD: ordering phases separately - number of decision boxes

	Modified Top-Down	Modified Depth-First	Modified Priority Depth-First	Depth-First, with Number of Leaves	Non-dynamic Top-Down Weights	Dynamic Top-Down Weights	Bottom-Up Weights	Event Criticality	Structural Importance
	1	2	3	4	5	6	7	8	9
Example 1	14	16	14	14	14	14	15	14	14
Example 2	91	150	99	99	92	114	197	91	91
Example 3	59	76	64	60	53	47	54	46	67
Example 4	89	151	119	107	87	79	20	54	
Example 5	469	506	472	472	472	480	290	378	477
Example 6	198	445	202	202	201	203	135	193	
Example 7	953	989	955	878	876	886	729	887	867
Example 8	258	889	575	750	159	266	103	148	
Example 9	336	1681	365	365	453	412	1208	357	
Example 10	298	2191	600	600	393	981	1413	248	
Example 11	1057	4862	4944	4044	675	1149	2695	400	

Table 7.2: CCD: ordering all mission - number of decision boxes

	Modified Top-Down	Modified Depth-First	Modified Priority Depth-First	Depth-First, with Number of Leaves	Non-dynamic Top-Down Weights	Dynamic Top-Down Weights	Bottom-Up Weights	Event Criticality
	1	2	3	4	5	6	7	8
Example 1	86	97	86	97	86	86	86	86
Example 2	189	247	189	247	189	189	247	189
Example 3	467	455	467	455	467	467	467	467
Example 4	238	625	238	625	238	238	625	238
Example 5	4545	7734	4545	7734	4545	4545	7731	4545
Example 6	3483	5491	3483	5491	3483	3483	3483	3482
Example 7	120947	211047	120947	211047	120947	120947	211047	120947
Example 8	25152	50157	25152	50157	25152	25152	25152	25152
Example 9	8694	43647	8642	43647	8694	8694	15311	8694
Example 10	209314	-	206270	-	209314	209314	209314	209314
Example 11	4692	15231	9438	15231	4692	4692	5488	4692

Table 7.3: BDD: ordering phases separately - number of non-terminal nodes

It can be seen that there is no ordering scheme that is best overall. For example, the worst ordering scheme for one example will be the best for another (see Bottom-up weights ordering scheme in Table 7.1: it produces one of the worst results for Example 2, but one of the best results for Example 8). But, as it can be noticed from the tables above, the size of cause-consequence diagram is considerable smaller that that of binary decision diagram. This can be seen comparing Tables 7.1 and 7.3 (i.e., Example 9, using modified top-down ordering scheme has 456 decision boxes in cause-consequence diagram, while using the BDD it has 8694 non-terminal nodes). In a few cases the binary decision diagram could not be produced as there was not enough computational capacity. This was for Example 10, which is a five phase system with fourteen components. The cause-consequence diagrams produced for those examples where considerably smaller. So, it can be concluded, that for the same phased mission systems, the cause-consequence diagram method is more efficient in terms of the size of diagram. Therefore the use of cause-consequence diagram method would help to reduce the computational resources needed for analysis of the phased mission systems. As for the best performing ordering scheme based on the few example used, on average, the event criticality performed well in both ordering components in each phase separately and ordering components for all mission at once. The second best one was non-dynamic top-down weights followed by modified top-down approach.

7.6 Analysis of phased mission system with multiple faults

For many systems a mission may be performed where multiple faults are possible. The outcomes will be different depending on the failure. An example of such a system would be an aircraft flight, where faults could be classified as minor, major or catastrophic. A minor fault would allow the flight to continue to the designated destination, a major fault would cause an emergency landing and a catastrophic fault would lead to loss of the aircraft.

This section considered an example of phased mission system with multiple faults. It is shown that phased missions with multiple faults can be analysed using cause-consequence diagrams, although it is more computationally intensive than in the single fault case.

Basic rules for creating a cause-consequence diagram for a phased mission system with multiple faults are outlined below:

1. Basic event transformation is applied to fault trees for the phase as for single fault case. Each fault needs to be represented by a different fault tree.

2. Separate cause-consequence diagrams are constructed for each fault in each phase. The same ordering scheme is used for each phase.

3. All failures for the same phase are combined into one cause-consequence diagram. This can be done by attaching the CCD for the next failure to each

outcome of the previous failure and removing repeated/inconsistent events. The consequences for each failure are noted in the consequence box.

4. Once the cause-consequence diagrams are obtained for each phase, they can be combined into one diagram for the whole mission. Each consequence box contains consequences for all failures within each phase. Again, this is done by attaching the CCD for next phase to each consequence of the previous phase (unless all failures considered occur in the previous phase) and then removing repeated/inconsistent events.

5. Quantification is performed using the standard quantification technique of cause-consequence analysis for phased mission systems. At this point, the cut sets and probabilities can be calculated for each fault separately.

7.6.1 Example

An example is used to illustrate a cause-consequence analysis method for phased missions when there are multiple failures possible in each phase. The example shown is of a three phased mission. The system is made up of three components and there are two faults to consider - minor and major. For example, in phase 1 if 'A' or 'B' fail then a minor failure occurs, but if in addition to 'A' or 'B' failing also fail 'C' or 'D', then a major failure occurs, such as loss of aircraft, etc. The fault trees for each failure in each phase are shown in Figure 7.11.

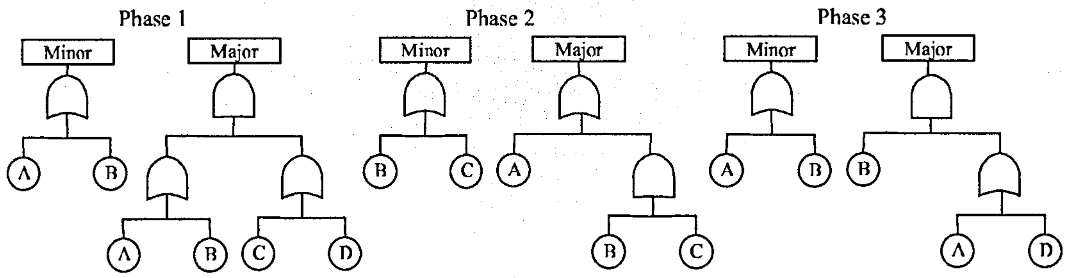


Figure 7.11: Fault trees for the mission

The first step of the algorithm requires basic event transformation to be performed to account for possible failures of the component in previous phases.

The next step is to construct cause-consequence diagrams for each failure in the phase. As this example is illustrating a three phased mission, this step will need to be repeated for each phase. In each following phase cause-consequence diagrams will get bigger as the number of basic events increases. The order in which components are considered is:

$$A < B < C < D$$

Cause-consequence diagrams for each failure in the phases are shown in Figures 7.12 to 7.15.

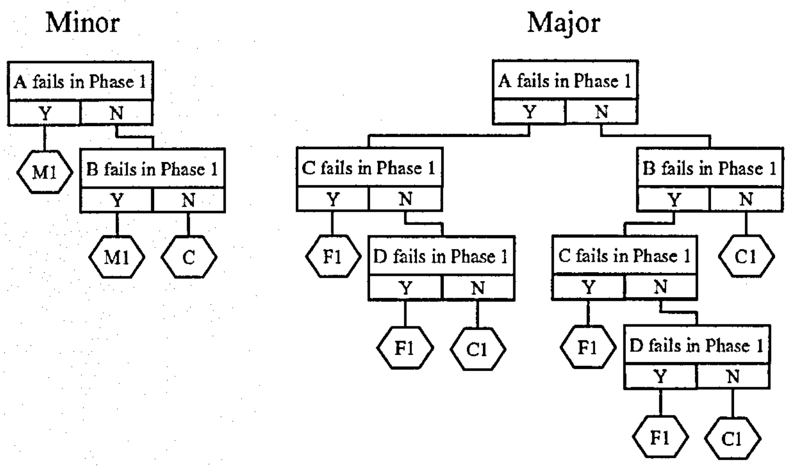


Figure 7.12: Phase 1 - cause-consequence diagrams for each failure

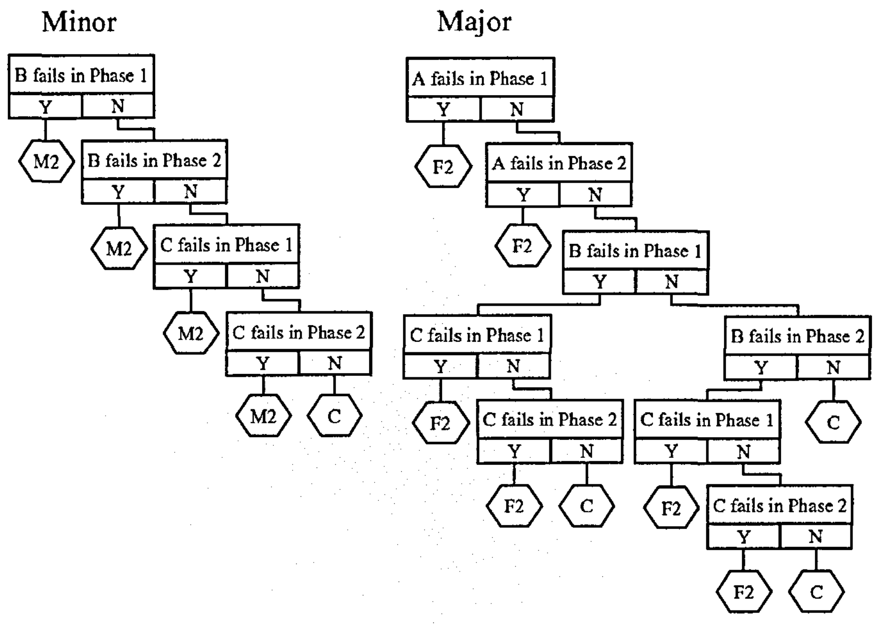


Figure 7.13: Phase 2 - cause-consequence diagrams for each failure

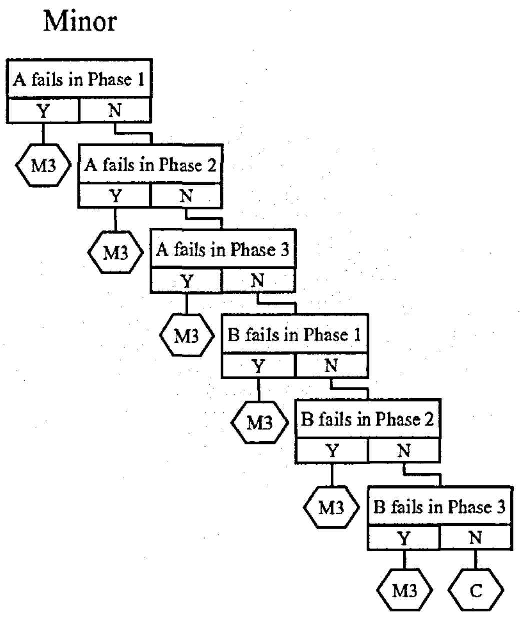


Figure 7.14: Phase 3 - cause-consequence diagrams for minor failure

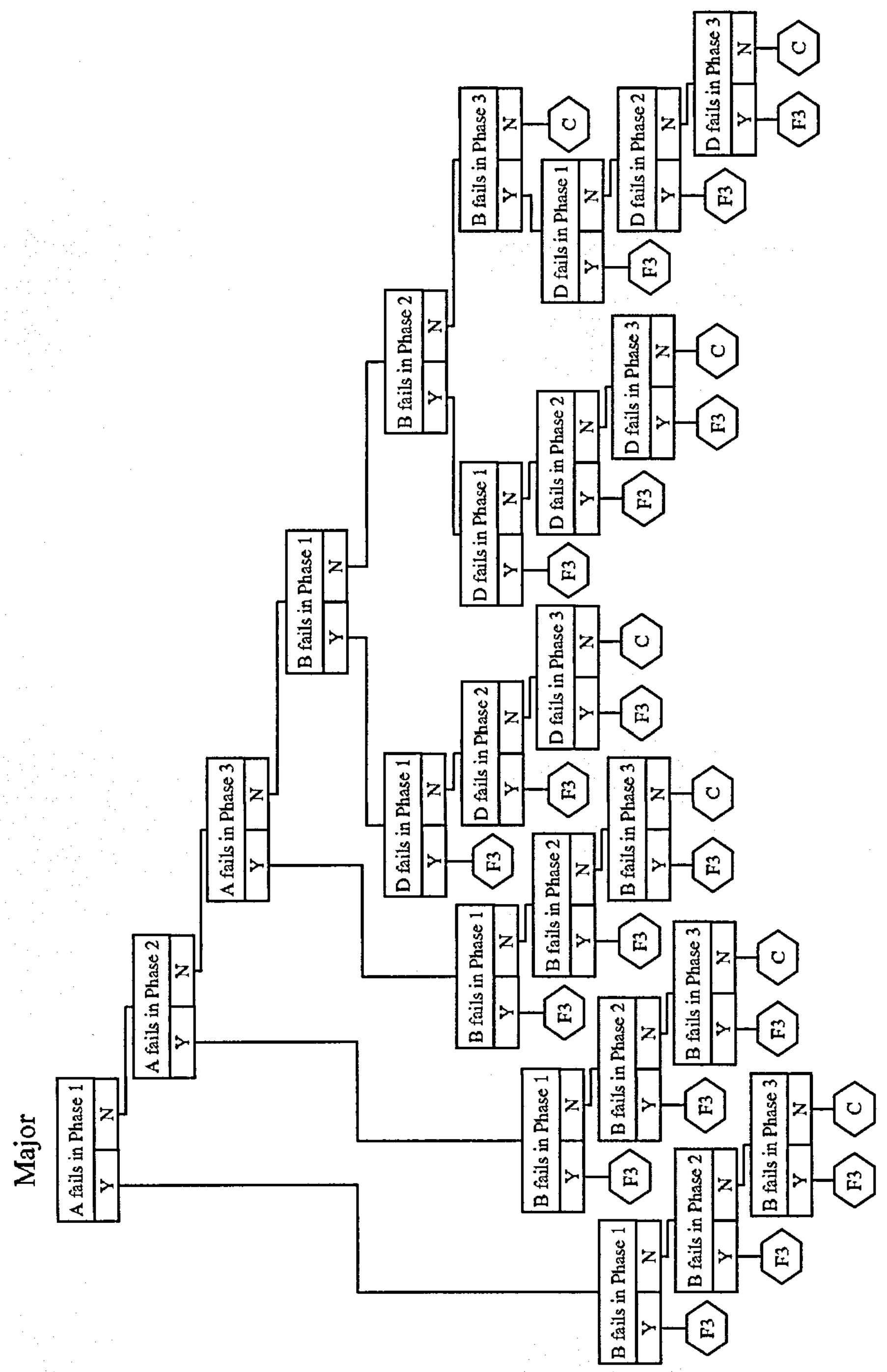


Figure 7.15: Phase 3 - cause-consequence diagrams for major failure

To obtain a combined cause-consequence diagram for the first phase, the cause-consequence diagrams for each failure need to be joined together. This is done by connecting the cause-consequence diagram for second fault to each outcome of the cause-consequence diagram for the first fault. In this example, the cause-consequence diagram for minor fault is attached to cause-consequence diagram for major fault. The outcome of this procedure is shown in Figure 7.16. The consequence boxes now show whether major or minor fault occurred or system succeeded in this phase.

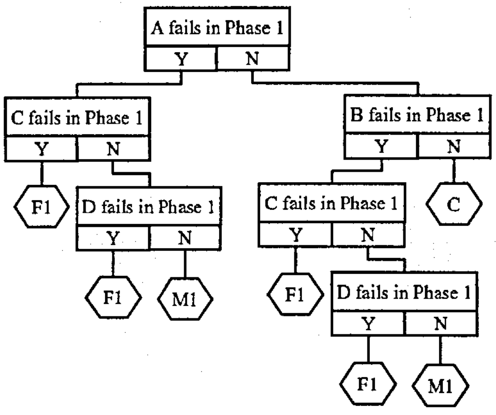


Figure 7.16: Phase 1 - combined cause-consequence diagram

In the same way the cause-consequence diagrams for each failure for phases 2 and 3 are combined into one diagram for each phase. These are shown in Figure 7.17 for phase 2 and Figure 7.18 for phase 3.

Once the cause-consequence diagrams are obtained for each phase, they can be combined in one diagram for the complete mission. The resulting cause-consequence diagram for this mission is shown in Figure 7.19. The final diagram is obtained by attaching the cause-consequence diagram for the following phase instead of the consequence boxes that state either system success in the particular phase or minor failure. Then the repeated and inconsistent events are removed from the diagram (for example, event 'A fails in phase 1' would be repeated in all three phases, but only one occurrence of this event is left in each path).

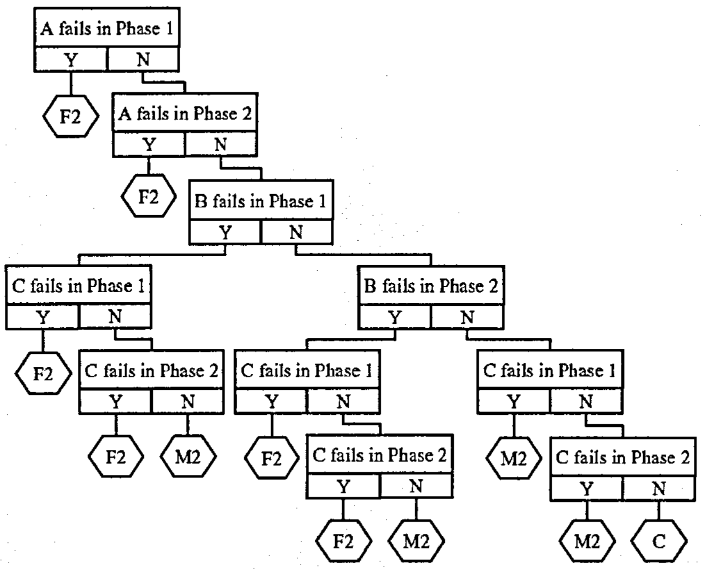


Figure 7.17: Phase 2 - combined cause-consequence diagram

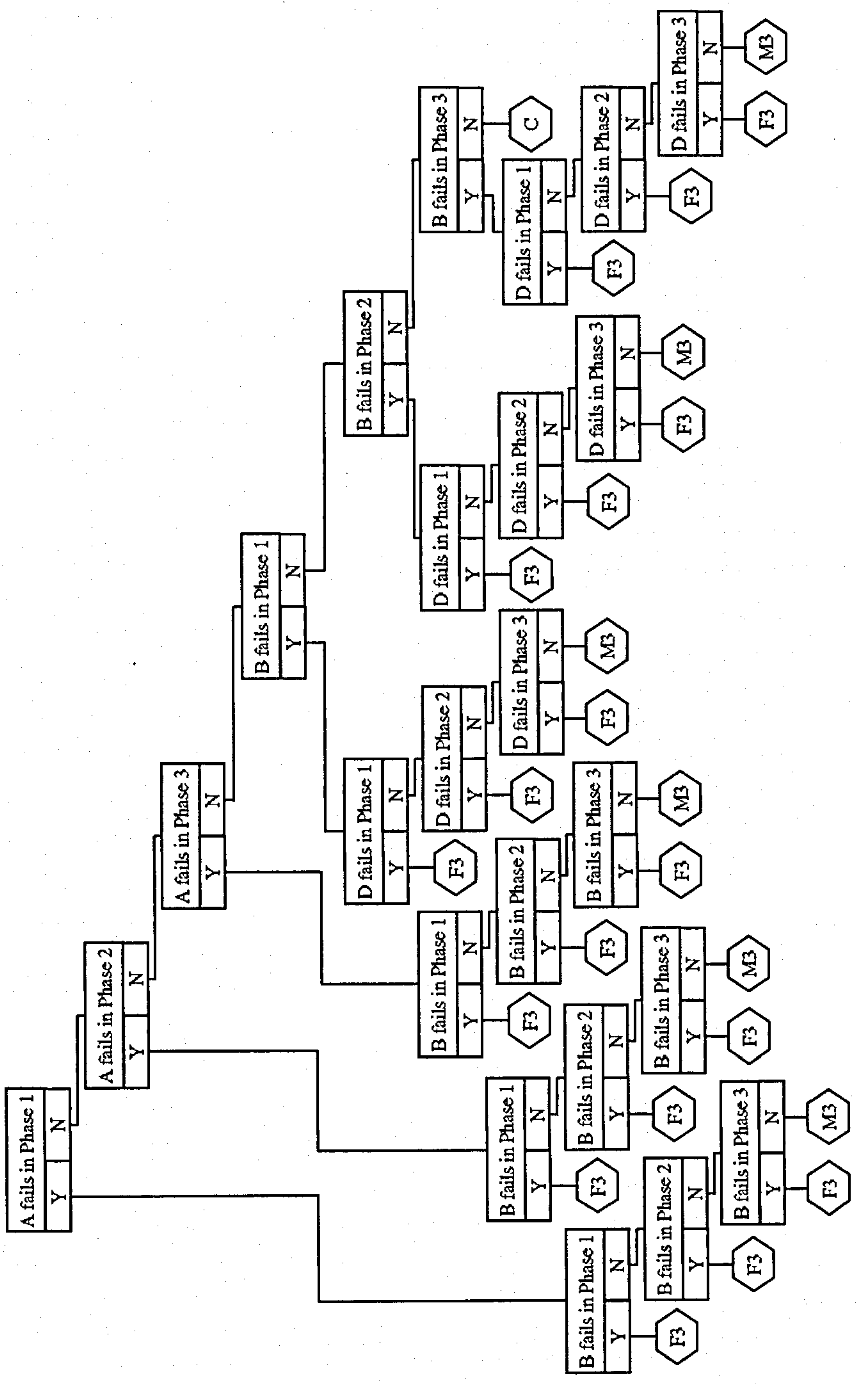


Figure 7.18: Phase 3 - combined cause-consequence diagram

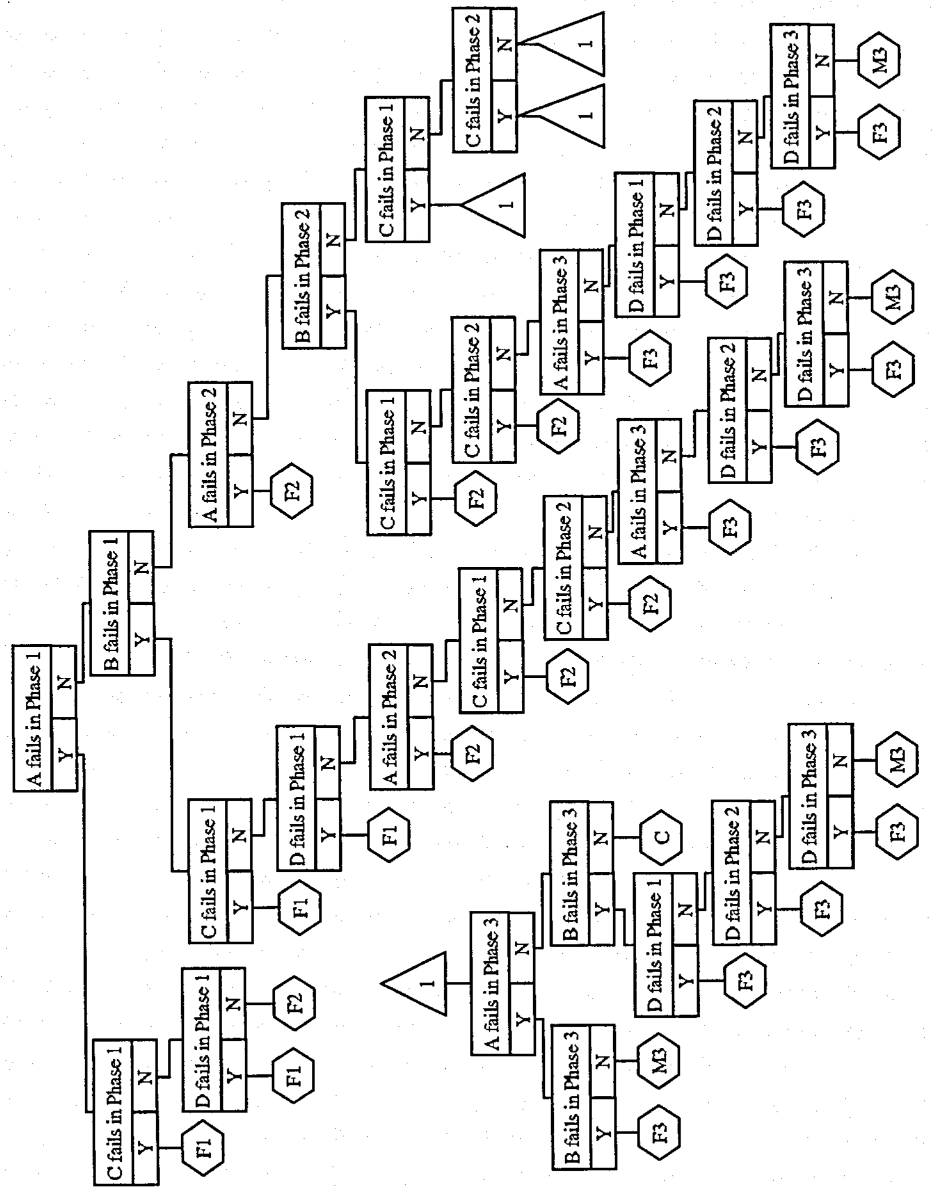


Figure 7.19: Combined cause-consequence diagram for a mission showing all failures

7.6.2 Quantitative and Quantitative analysis

Qualitative analysis in case of multiple faults is performed the same way as described in Section 7.3.2, just in this case instead of determining list of causes for phase failures, these lists can be obtained for each fault in each phase. The lists are simplified in the same way as before.

The quantitative analysis is performed the same way as in Section 7.3.3, but instead of calculating the probability of each phase, the result can be obtained for each fault in different phases.

7.7 Discussion

As it has been shown in this section, cause-consequence diagram method can be successfully implemented for the analysis of phased mission systems. This method is superior over binary decision diagrams as it can represent a whole system in one diagram without loosing valuable information about phase failures. Also, as the diagram is easy to follow and contains descriptions of components states, it can be presented to those without much prior knowledge in risk analysis.

It has also been shown here that in the same way as with the binary decision diagrams, the ordering of the components can influence the size of the diagram hugely and the selection of an effective variable ordering scheme can produce a very efficient cause-consequence diagram for the phased mission problem.

Two methods were presented for the construction of the cause-consequence diagram for phased mission. For this project, method 1 as presented earlier was implemented as a computer code for extraction of the results.

Although the methods shown above offer different ways of constructing the cause-consequence diagram, the quantification procedure is the same for both methods. It has also been shown that as the cause-consequence diagram does not loose system state information when moving on to the next phase, it can be used to model not only single fault systems, but also those, which can fail in different ways.

8. MODELLING AIRCRAFT FLIGHT USING THE CAUSE-CONSEQUENCE ANALYSIS

8.1 Introduction

This chapter presents an example of the application of the cause-consequence diagram method to a non-repairable phased mission system. The system considered is an aircraft flight consisting of six phases: take off, climb, cruise, descent, approach and landing. Each phase uses differing functional elements of the system and so the causes of failure in each phase are different. Using an automated approach developed previously to construct and quantify the cause-consequence diagram the causes of mission failure are identified and investigated.

8.2 Aircraft flight system

To illustrate the use of the CCD method for complex systems an aircraft flight has been considered. The flight considered will consist of 6 phases: take off, climb, cruise, descent, approach and landing. A diagram of the flight is shown in Figure 8.1.

Each phase of the flight may utilise different systems within the aircraft. For example, the landing gear is required only in the climb and landing phases. The aircraft systems that were used in this model are:

- Propulsion system, which consists of:
  - Gas turbine engine. A 2 engine aircraft is considered.
  - Thrust. The control of engine thrust is achieved by altering the fuel flow to the combustion chamber of the engine. If the fuel flow is increased, the resulting gas temperature is higher. The higher temperature means that more thermal energy conversion can take place at the turbines resulting in an increase in speed. As the turbines are directly linked to their relative

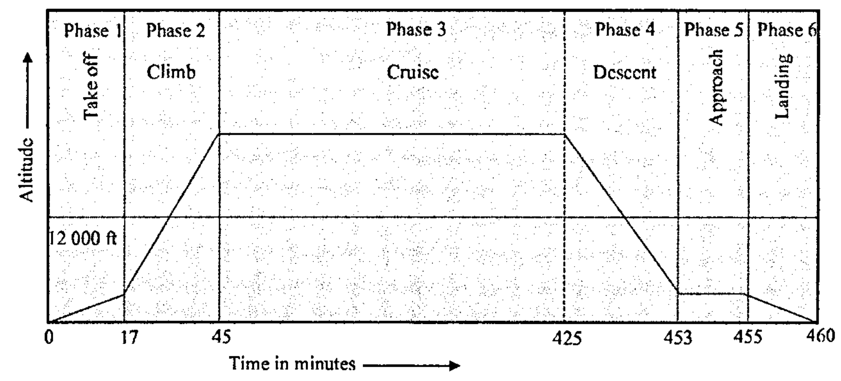


Figure 8.1: Aircraft Flight Diagram

compressors, an increase in compressor speed occurs with a resultant increase in the engine mass flow and hence thrust.

- Flight control system. This can be considered as 2 subsystems.
  - Primary flight control. The aim of primary flight control is to control manoeuvre of the aircraft about each of its axes.
  - Secondary flight control. The secondary flight control is used intermittently to change the value of lift and drag generated by the aircraft surfaces, but not affect the trajectory.
- Fuel system. The aim of the fuel system is to provide an effective means of replenishment, storage and fuel feed to the engines under all anticipated operating conditions. It can be considered as 2 subsystems:
  - Fuel feed which provides a flow of fuel to the engines under all anticipated flight conditions.
  - Fuel transfer which provides a means of fuel transfer between tanks and via external sources in flight and during ground situations.
- Pneumatic supply. To enable main engine starting from a variety of supply air sources an integrated pneumatic system is incorporated into the aircraft design. This system not only facilitates starting, but also supplies low pressure air for other aircraft services such as anti-icing and water tank pressurisation

- Hydraulic system. Many of the aircraft services are powered by hydraulic power including flight control systems, flaps, retractable undercarriages and wheel brakes. It has the advantage of transmitting high forces rapidly and accurately along the lightweight pipes of any size, shape and length.

- Environmental control system. The environmental control systems overall aims are to minimise the risk to the safe operation of the aircraft in all anticipated weather and operational conditions and to ensure the safety and comfort of the people on aboard. It can be considered to be made up of the 2 subsystems:

  - Anti-icing system. The anti-icing system consists of ice sensing and ice removal. Most severe icing occurs when a cold aircraft descends from high altitude through rain or cloud which is already below zero ambient temperatures. The effects of ice may include increase in mass, loss of lift (small accretion can reduce lift by 30%), increase in drag (ice formation can increase drag by 200%), jammed controls, obscured vision, loss of engine power (intake restriction), etc.

  - Cabin control. At an altitude of about 9000m, typical of an aircraft cruising altitude, ambient air pressure is 300.9 millibars and temperature -44.44°C. Humans feel comfortable with air pressure at 1013.2 millibars and temperature 20°C. To reproduce these conditions at high altitude would necessitate an extremely strong and heavy structure. As a compromise 2348m is the recognised maximum normal internal cabin altitude for air transport aircraft. This altitude would provide safe and comfortable levels of oxygen and require a lighter structure than the one of sea level as the maximum differential would be only 451.8 millibars compared with the one of sea level (712.3 millibars). The cabin conditioning system has also to compensate for the low ambient temperature and bring the internal cabin temperature into a range that is comfortable for humans. In summary the subsystem has to provide safe oxygen levels, ensure comfortable cabin temperature, control humidity and provide freshness.

- Landing gear. Landing gear is defined as those components necessary to enable take-off and landing to be carried out safely. It excludes thrust reversal and control surfaces. Two failure modes have been considered:

  - Retraction - up
  - Retraction - down

- Flight navigation system. This is made up of 4 subsystems:
  - INS. An inertial navigation system which measures the position and altitude of a vehicle by measuring the accelerations and rotations applied to the system's inertial frame. The system is aligned to start up with known coordinates. It is widely used because it refers to no real-world item beyond itself. It is therefore immune to jamming and deception.
  - GPS. Global Positioning System (GPS) is an accurate means of providing continuous worldwide navigation information using a system of satellites. The system consists of three main components: control, space and user. GPS provides highly accurate positional, velocity and time data.
  - VOR/ADF/DME. VOR (VHF Omni-Range) is an internationally recognised short-range navigation aid. The usable VOR range varies with aircraft altitude but is effective up to about 300 miles. The principle of operation is based on a ground-based transmitter providing radial signal output, this is received by equipment on the aircraft to provide bearing information to an identified beacon. Automatic Direction Finding (ADF) function identifies bearing to a beacon to which the aircraft receiver is tuned. Distance Measuring Equipment (DME) is often integrated with VOR to provide range and bearing information. Initially the aircraft transmits a signal to the ground receiver which then responds sending a signal to the aircraft. The lapsed time difference from transmission to reception provides the data to calculate the slant range from aircraft to ground station.
  - ILS/MLS. Instrument Landing System (ILS)/Microwave Landing System (MLS) is a landing aid providing both lateral and vertical guidance. Lateral guidance is provided by a VHF localiser aerial and vertical (descent slope) by a UHF glideslope aerial. The development and adoption of the MLS provides improvement in the quality of guidance

over ILS. The system has a wider scope of view and higher scanning rates. This provides increased data with which to control the aircraft.

- Electrical system. The primary function of an aircraft electrical system is to generate, regulate and distribute electrical power throughout the aircraft. The aircraft electrical power system is used to supply power for services including lighting, avionics, fuel system booster pumps and valves, control of hydraulic system components, in-flight entertainment, flight control systems and aircraft environmental control. Essential power is power that the aircraft needs to be able to continue safe operation.

The aircraft flight chosen was taken to be one that included flying over the ocean (i.e. London to New York). Also, for modelling purposes it was decided to consider a twin-engine aircraft (i.e. Boeing 777). Redundancy is not taken into account in this example, except for the engines. It was assumed that all components are non-repairable whilst in flight.

The failure modes considered and notation adopted are shown in Table 8.1. The failures considered in this example would be catastrophic. The failure frequencies used in this example are only for modelling purposes and may not necessarily represent the real system.

System	Subsystem	Failure mode description	Notation	Failures /million hours
Propulsion	Engine	Failure of engine 1	E1	0.999
		Failure of engine 2	E2	0.999
	Thrust	Failure of thrust to engine 1	T1	0.295
		Failure of thrust to engine 2	T2	0.295
		Failure of thrust reverser	TR	0.467
Flight control system	Primary flight control	Failure of primary flight control	PFCF	0.147
	Secondary flight control	Failure of secondary fight control	SFCF	0.139

Fuel system	Fuel feed	Failure of fuel feed	FFF	0.393
	Fuel transfer	Failure of fuel transfer	FTF	0.393
System	Subsystem	Failure mode description	Notation	Failures /mhours
Pneumatic supply	Pneumatic supply	Failure of pneumatic supply	P	0.288
Hydraulic system	Hydraulic system	Failure of hydraulic system	HS	0.446
Environmental control system	Anti-icing system	Failure of ice sensing	AIS	0.086
		Failure of ice removal	AIR	0.131
	Cabin control system	Failure of pressure control	CCPC	0.139
		Failure of temperature control	CCTC	0.148
Landing gear	Landing gear	Failure of landing gear retraction up	LGU	0.205
		Failure of landing gear retraction down	LGD	0.205
Flight navigation system	Flight navigation system	Failure of INS	INS	0.0257
		Failure of GPS	GPS	0.237
		Failure of VOR/ADF/DME	VOR	0.309
		Failure of ILS/MLS	ILS	0.0026
Electrical system	Electrical system	Failure of power	ESP	0.0676

Table 8.1: Failure rates for the subsystems

8.2.1 Phase 1

Phase 1, the take off phase, will be completed successfully, if all systems required in this phase operate successfully. These systems are: propulsion system (both engines have to function properly), flight control, fuel feed, pneumatic supply, hydraulic system and anti-icing system. The fault tree for failure in phase 1 is shown in Figure 8.2 which has minimal cut sets shown in Table 8.2.

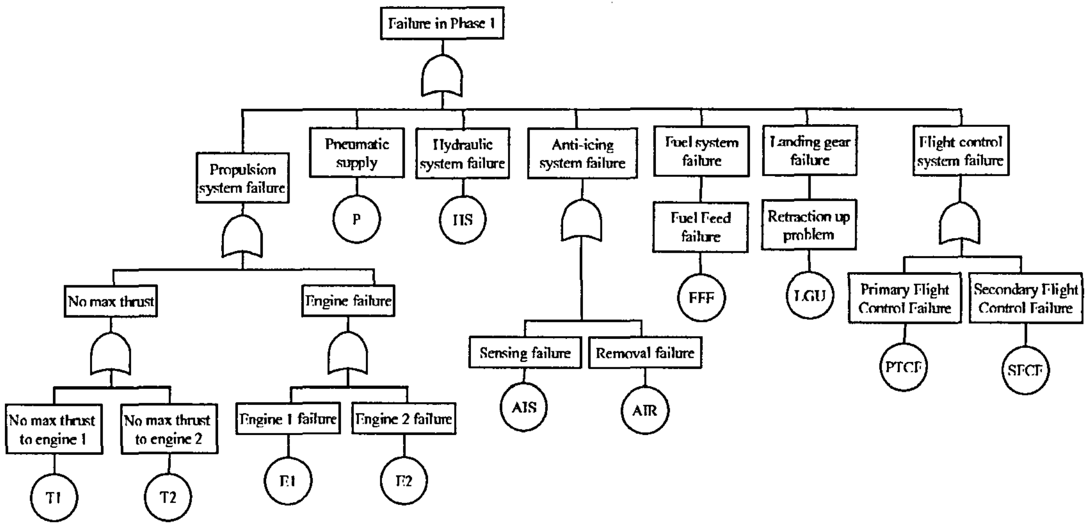


Figure 8.2: Fault Tree for Phase 1

SFCF	E2	FFF	P
AIR	E1	T2	PFCF
AIS	HS	T1	

Table 8.2: Minimal cut sets for Phase 1

8.2.2 Phase 2

For phase 2, the climb phase, propulsion, hydraulic and pneumatic systems are required to work as in phase 1. Also required are: fuel transfer, primary flight control, environmental control, the flight navigation systems GPS, VOR/ADF/DME and landing gear. The fault tree for failure in phase 2 is shown in Figure 8.3 which has minimal cut sets shown in Table 8.3.

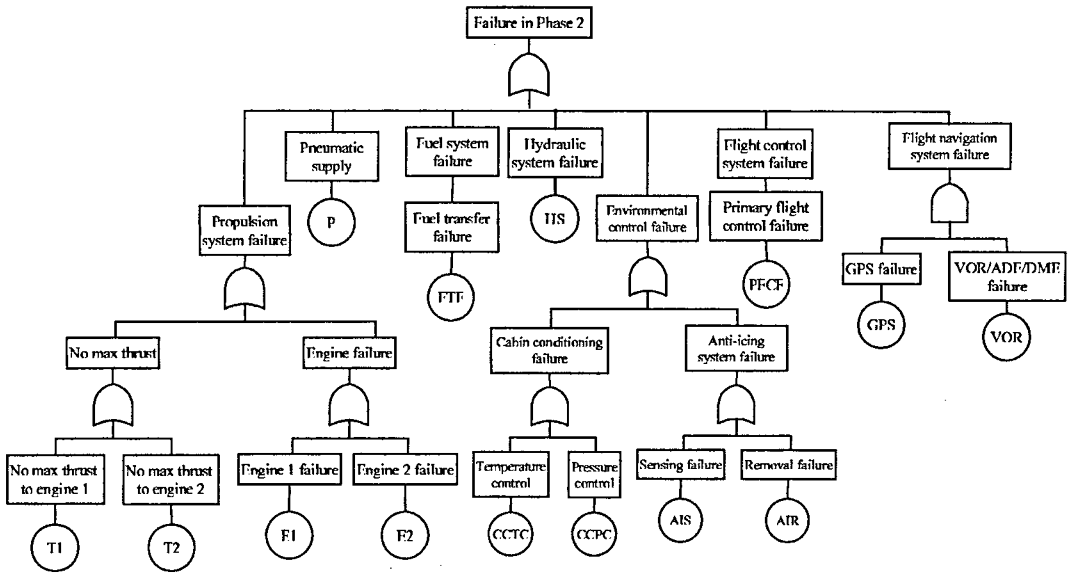


Figure 8.3: Fault Tree for Phase 2

AIR	E1	T1	LGU	VOR
AIS	HS	T2	PFCF	CCPC
E2	FTF	P	GPS	CCTC

Table 8.3: Minimal cut sets for Phase 2

8.2.3 *Phase 3*

During cruise, propulsion system failure will be caused by the failure of both engines which will lead to aircraft failure. The thrust wouldn't cause critical failure in this phase. Fuel feed, flight control, hydraulic system and pneumatic supply are required to work the same way as in phase 2.

Anti-icing system failure wouldn't be critical during cruise, but cabin control failure would as it is necessary to keep cabin pressure and temperature within a comfortable range. Failure of the navigation systems GPS or INS would lead to failure in this phase as would electrical system failure (power failure). The fault tree for failure in phase 3 is shown in Figure 8.4 which has minimal cut sets shown in Table 8.4.

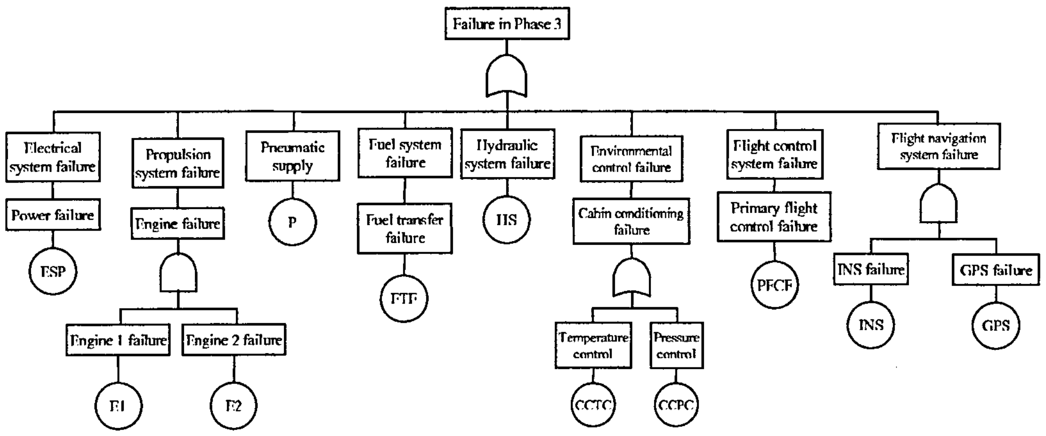


Figure 8.4: Fault Tree for Phase 3

HS	P	E1 · E2	GPS	CCTC
FTF	PFCF	ESP	INS	CCPC

Table 8.4: Minimal cut sets for Phase 3

8.2.4 Phase 4

Failure in phase 4, the descent phase, has causes similar to failure in the climb phase, phase 2, as shown in the fault tree in Figure 8.5. Failure will occur if the pneumatic, hydraulic, fuel, flight control and environmental control systems fail in the same way as in phase 2. The propulsion system failure will be caused by the failure of both engines which will lead to aircraft failure. The thrust wouldn't cause critical failure in this phase. The failure of the GPS navigation system will also lead to phase failure. The minimal cut sets for this phase are shown in Table 8.5.

AIR	HS	E1 · E2	P	CCPC
AIS	FTF	PFCF	CCTC	GPS

Table 8.5: Minimal cut sets for Phase 4

8.2.5 Phase 5

During approach, phase 5, the propulsion system, pneumatic supply and hydraulic system are required to work as in the previous phase. For environmental control - only the anti-icing system failure would be critical. Flight control system will again depend on both primary and secondary flight controls - failure of any of them would be critical. During descent flight navigation needs GPS, VOR/ADF/DME and ILS/MLS and hence failure of any of them will lead to phase failure. The fault tree for failure in phase 5 is shown in Figure 8.6 which has minimal cut sets shown in Table 8.6.

SFCF	AIS	$E1 \cdot E2$	P	GPS
AIR	HS	PFCF	ILS	VOR

Table 8.6: Minimal cut sets for Phase 5

8.2.6 Phase 6

Failure will occur in the landing phase if the hydraulic system and flight control system fail. Also failure of the flight navigation system ILS/MLS and the landing gear is critical. In addition, the electrical system and propulsion system (both engines and thrust reverser) are required to work. See the fault tree in Figure 8.7

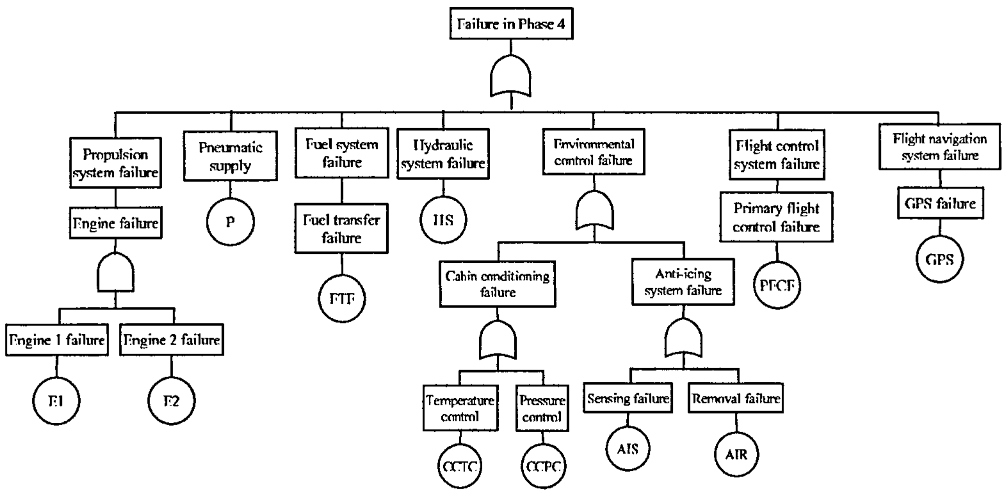


Figure 8.5: Fault Tree for Phase 4

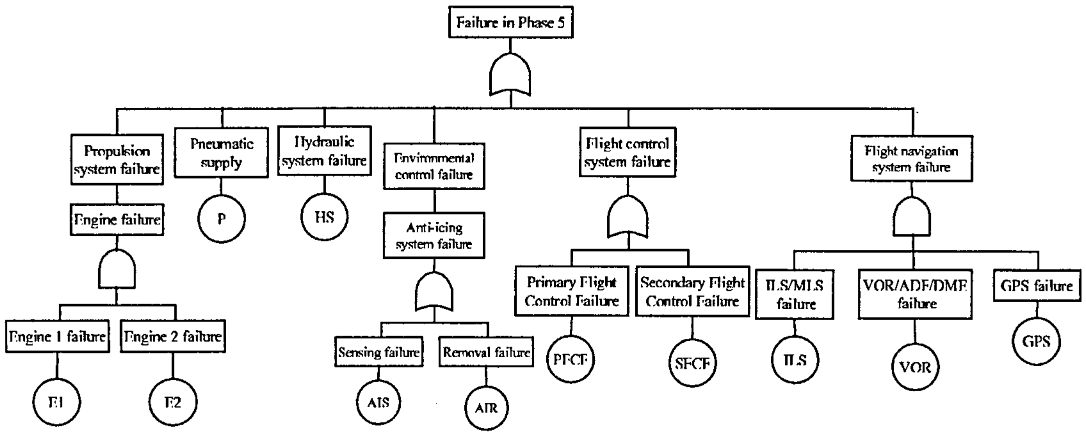


Figure 8.6: Fault Tree for Phase 5

with minimal cut sets given in Table 8.7.

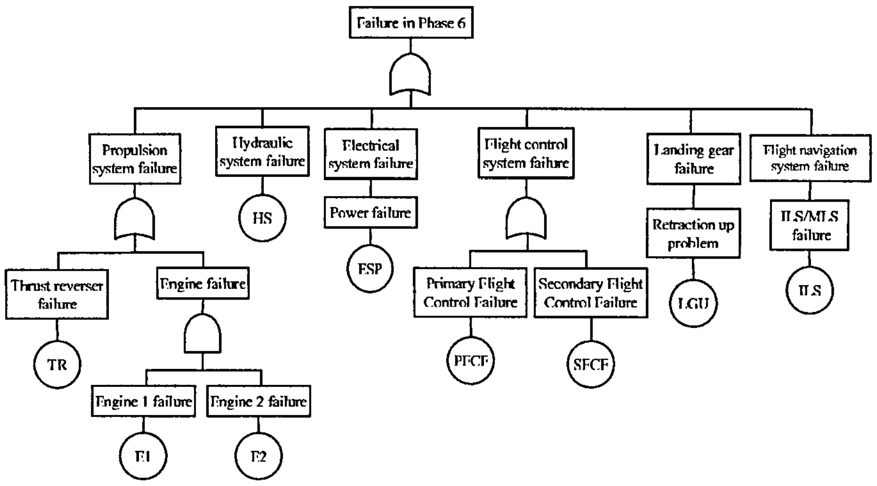


Figure 8.7: Fault Tree for Phase 6

SFCF	PFCF	E1 · E2	ESP
LGD	HS	ILS	TR

Table 8.7: Minimal cut sets for Phase 6

8.3 Construction of the Cause-Consequence Diagram

In order to construct the cause-consequence diagram it is necessary to assume a certain order in which the component failure events will be considered. For this example we assume the following order:

$FFF_1 < P_1 < HS_1 < T_1 < T2_1 < E1_1 < E2_1 < PFCF_1 < SFCF_1 < AIS_1 < AIR_1 < LGU_1 < LGU_2 < FTF_1 < FTF_2 < PFCF_2 < P_2 < HS_2 < E1_2 < E2_2 < CCTC_1 < CCTC_2 < CCPC_1 < CCPC_2 < AIS_2 < AIR_2 < GPS_1 < PS_2 < VOR_1 < VOR_2 < ESP_1 < ESP_2 < ESP_3 < FTF_3 < PFCF_3 < HS_3 < P_3 < E1_3 < E2_3 < CCTC_3 < CCPC_3 < GPS_3 < INS_1 < INS_2 < INS_3 < P_4 < HS_4 < FTF_4 < GPS_4 < E1_4 < E2_4 < CCTC_4 < AIS_3 < AIS_4 < AIR_3 < AIR_4 < P_5 < HS_5 < E1_5 < E2_5 < AIS_5 < AIR_5 < PFCF_5 < SFCF_2 < SFCF_3 < SFCF_4 < SFCF_5 < GPS_5 < VOR_3 < VOR_5 < ILS_1 < ILS_2 < ILS_3 < ILS_4 < ILS_5 < ESP_4 < ESP_5 < ESP_6 < ILS_6 < LGD_1 < LGD_2 < LGD_3 < LGD_4 < LGD_5 < LGD_6 < HS_6 < TR_1 < TR_2 < TR_3 < TR_4 < TR_5 < TR_6 < E1_6 < E2_6 < PFCF_6 < SFCF_6$

Where the subscripts refer to the phase in which the failure occurs. The program allows the order to be entered manually, or to automatically generate it from the fault trees. The order of events shown above is automatically generated by the program, where the components are considered in the order as they appear in fault tree data files.

Following the order of component failures given, each component failure event is added to the diagram as a decision box one by one, as described in cause-consequence diagram construction method 1 (see Section 7.3.1). For this example, consequences F1, F2, F3, F4, F5 and F6 mean mission failure due to system failure in phase 1, 2, 3, 4, 5 or 6 respectively, C represents mission success.

The resulting diagram output by the program is represented in a list form. Each decision/consequence box is assigned a number and the program lists the numbers of the decision/consequence boxes it is pointing to on the YES and NO branches. For each box, the program also gives the number of the previous box in the branch.

8.4 Analysis of the Cause-Consequence Diagram

In order to quantify the resulting cause-consequence diagram and obtain the probability of mission failure it is necessary to input data for the subsystems.

No real data for the system was obtained, and data has been generated which was believed to be realistic for the subsystems considered. The data is included in Table 1.

The durations for the phases were taken to be those of a London (Heathrow) - New York (J.F. Kennedy) flight and are shown in Table 8.8.

Phase 1	17 min
Phase 2	28 min
Phase 3	380 min
Phase 4	28 min
Phase 5	2 min
Phase 6	5 min

Table 8.8: Phase duration times

8.4.1 Qualitative analysis

Conditions causing any outcome event (implicants) are established by investigating each decision box on the path to the outcome and listing the component failure or success in the phase as indicated by the exit path from the decision box. In the example considered there are 375 failure outcomes. The component conditions for each of these outcomes were determined and some of them are listed below. The notation used for the events is that given in Table 8.1. Numbers in front of the implicant indicate the number of the consequence box in the cause-consequence diagram.

F1 - Mission failure in Phase 1

2. FFF_1

4. $P_1 \wedge \overline{FFF_1}$

6. $HS_1 \wedge \overline{P_1} \wedge \overline{FFF_1}$

8. $T1_1 \wedge \overline{HS_1} \wedge \overline{P_1} \wedge \overline{FFF_1}$

10. $T2_1 \wedge \overline{T1_1} \wedge \overline{HS_1} \wedge \overline{P_1} \wedge \overline{FFF_1}$

12. $E1_1 \wedge \overline{T2_1} \wedge \overline{T_1} \wedge \overline{HS_1} \wedge \overline{P_1} \wedge \overline{FFF_1}$

14. $E2_1 \wedge \overline{E1_1} \wedge \overline{T2_1} \wedge \overline{T_1} \wedge \overline{HS_1} \wedge \overline{P_1} \wedge \overline{FFF_1}$

16. $PFCF_1 \wedge \overline{E2_1} \wedge \overline{E1_1} \wedge \overline{T2_1} \wedge \overline{T_1} \wedge \overline{HS_1} \wedge \overline{P_1} \wedge \overline{FFF_1}$

18. $SFCF_1 \wedge \overline{PFCF_1} \wedge \overline{E2_1} \wedge \overline{E1_1} \wedge \overline{T2_1} \wedge \overline{T_1} \wedge \overline{HS_1} \wedge \overline{P_1} \wedge \overline{FFF_1}$

20. $AIS_1 \wedge \overline{SFCF_1} \wedge \overline{PFCF_1} \wedge \overline{E2_1} \wedge \overline{E1_1} \wedge \overline{T2_1} \wedge \overline{T_1} \wedge \overline{HS_1} \wedge \overline{P_1} \wedge \overline{FFF_1}$

22. $AIR_1 \wedge \overline{AIS_1} \wedge \overline{SFCF_1} \wedge \overline{PFCF_1} \wedge \overline{E2_1} \wedge \overline{E1_1} \wedge \overline{T2_1} \wedge \overline{T_1} \wedge \overline{HS_1} \wedge \overline{P_1} \wedge \overline{FFF_1}$

Implicants for the later phases contain more events due to consideration of the earlier phases. For example, one implicant for phase 2 is:

F2 - Mission failure in Phase 2

25. $LGU_2 \wedge \overline{LGU_1} \wedge \overline{AIR_1} \wedge \overline{AIS_1} \wedge \overline{SFCF_1} \wedge \overline{PFCF_1} \wedge \overline{E2_1} \wedge \overline{E1_1} \wedge \overline{T2_1} \wedge \overline{T_1} \wedge \overline{HS_1} \wedge \overline{P_1} \wedge \overline{FFF_1}$

Each failure mode in the list contains a progression of states for the same component. For example, outcome 25 has component *LGU* working throughout phase 1 and then failing in phase 2. After simplification outcome 25 would be:

25. $LGU_2 \wedge \overline{AIR_1} \wedge \overline{AIS_1} \wedge \overline{SFCF_1} \wedge \overline{PFCF_1} \wedge \overline{E2_1} \wedge \overline{E1_1} \wedge \overline{T2_1} \wedge \overline{T_1} \wedge \overline{HS_1} \wedge \overline{P_1} \wedge \overline{FFF_1}$

Considering just the failed states for the systems which lead to mission failure gives minimal cut sets. Some of the minimal cut sets leading to phase failures are shown in Table 8.9. The notation used for the events is that given in Table 8.1. These minimal cut sets are obtained after basic event approximation was applied to fault trees of each phase. The minimal cut sets in Table 8.2 to Table 8.7 list minimal cut sets before basic event transformation is applied to fault trees.

Phase 1	Phase 2	Phase 3
AIR_1	AIR_2	$CCPS_3$
AIS_1	AIS_2	$CCTC_3$
$E1_1$	$CCPC_1$	$E1_3 \wedge E2_3$
$E2_1$	$CCPC_2$	ESP_1
FFF_1	$CCTC_1$	ESP_2
HS_1	$CCTC_2$	ESP_3
P_1	$E1_2$	FTF_3
$PFCF_1$	$E2_2$	GPS_3
$SFCF_1$	FTF_1	HS_3
$T1_1$	FTF_2	INS_1
$T2_2$	GPS_1	INS_2
	GPS_2	INS_3
	HS_2	P_3
	LGU_1	$PFCF_3$
	LGU_2	
	P_2	
	$PFCF_2$	
	VOR_1	
	VOR_2	

Table 8.9: Minimal cut sets

Consequence	Probability (exact)	Probability (coherent approximation)	Difference
Failed in phase 1	0.00028335	0.00028336	-0.00000001
Failed in phase 2	0.00050429	0.00050447	-0.00000018
Failed in phase 3	0.00087139	0.00087238	-0.00000099
Failed in phase 4	0.00015541	0.00015566	-0.00000025
Failed in phase 5	0.00022142	0.00022183	-0.00000041
Failed in phase 6	0.00040186	0.00040272	-0.00000086
Completed	0.99756235	0.99755957	0.00000278

Table 8.10: Mission results

8.4.2 Quantitative analysis

The results for the probability of mission failure and success have been obtained using the developed program for phased mission analysis and are shown in Table 8.10.

As the system investigated is very reliable, the coherent approximation results are very close to exact results with the difference starting in the 6th decimal place.

For the generated data the probability of the aircraft completing flight successfully was found to be 0.997562. As no real life data or result were obtained, it is not possible to compare the result of the modelling with the real life.

8.5 Conclusions

In this chapter an example of aircraft flight was presented. Although the system in reality is much more complex, it gives an overview of the capability of the cause-consequence diagram and demonstrates that cause-consequence diagram methods can be applied to real life non-repairable systems.

In this example fault trees were constructed for each phase to illustrate system behaviour. The computer code implementing method 1 for the construction of the cause-consequence diagram was used to obtain the results. The results obtained could not be compared with the real life results as no real data was available for the system and also because of the simplified approach to the example. Exact and coherent approximate quantification techniques were used on the cause-consequence diagram, both of them producing very close results. This was not surprising for this example as coherent approximation usually produce results close to the exact calculation for very reliable systems.

9. MODULARISATION OF PHASED MISSION SYSTEMS

9.1 Introduction

In order to reduce the complexity of a fault tree, modularisation techniques can be applied. One such technique identifies independent subtrees within the fault tree, which are called modules. A module is an independent section of a fault tree with no inputs that appear anywhere else in the tree and no outputs to the rest of fault tree except from its output event. The advantage of the modularisation is that each module can be regarded as an individual fault tree and analysed independently.

There are several modularisation techniques available for recognising fault tree modules. The modularisation technique introduced in this work is the linear-time algorithm [26]. It is later applied to the cause consequence diagrams for phased mission systems to see if it would be beneficial for reducing complexity of the larger systems.

9.2 The Linear-Time Algorithm

The modules of the fault tree are identified after two depth-first traversals of it. The first traversal records numbers of step-by-step visits for each gate and event: the step number at the first, second and final visits to that node. The second traversal through the fault tree finds the maximum of the last visits and the minimum of the first visits to the descendants of each gate.

To illustrate the procedure, the fault tree in the Figure 9.1 is used.

Starting at the top event the depth-first traversal visits each gate and event and the order in which they are visited is shown in Table 9.1. Each gate is visited at least twice: first time on the way down the fault tree and once more on the way back up the fault tree. If the gate is visited once already, then on the second visit to the gate its inputs are not visited. This can be noticed in step 23 in Table 9.1 where gate 'G3' is visited for the second time, but its descendants are not re-visited.

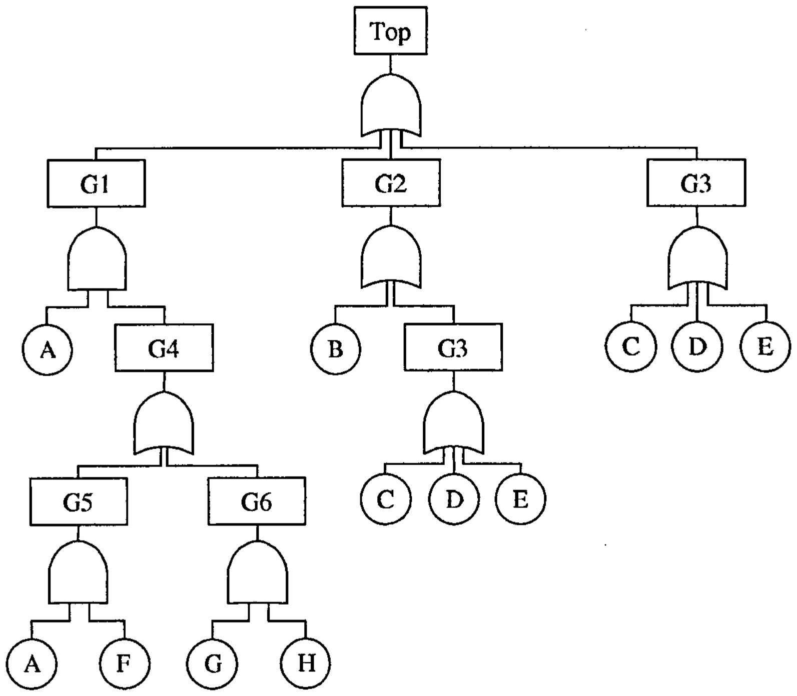


Figure 9.1: Example fault tree to demonstrate the linear-time algorithm

Step number	1	2	3	4	5	6	7	8	9	10	11	12
Node	Top	G1	A	G4	G5	A	F	G5	G6	G	H	G6

Step number	13	14	15	16	17	18	19	20	21	22	23	24
Node	G4	G1	G2	B	G3	C	D	E	G3	G2	G3	Top

Table 9.1: First traversal through the fault tree

Gate	Top	G1	G2	G3	G4	G5	G6
First visit	1	2	15	17	4	5	9
Second visit	24	14	22	21	13	8	12
Last visit	24	14	22	23	13	8	12
Max	23	13	23	20	12	7	11
Min	2	3	16	18	3	3	10

Table 9.2: Step numbers for gates

Table 9.2 and Table 9.3 show step numbers for the first, second and last visits to each gate and event respectively.

Gate	A	B	C	D	E	F	G	H
First visit	3	16	18	19	20	7	10	11
Second visit	6	16	18	19	20	7	10	11
Last visit	6	16	18	19	20	7	10	11

Table 9.3: Step numbers for events

The second traversal through the fault tree determines the maximum of the last visits (max) and minimum of the first visits to the descendants (min) of each gate. These results are shown in Table 9.2. If any descendants of the gate have a first visit step number smaller than the first visit step number of the gate, then it must have occurred beneath some other gate and therefore this gate cannot be a module. In the same way, if the last visit to the any descendant of the gate occurred later than the second visit to the gate, then this descendant must occur somewhere else in the fault tree and the gate again cannot be identified as a module. Therefore, the gate can be identified as a module if and only if it satisfies both conditions:

1. The first visit to each descendant is after the first visit to the gate.

2. The last visit to each descendant is before the second visit to the gate.

This ensures that none of the descendants of a particular gate can appear anywhere else in the fault tree, except beneath another occurrence of the same gate.

From Table 9.2 can be noticed, that gates 'G2', 'G4' and 'G5' cannot be modules. Gate 'G2' does not satisfy the second condition as the maximum of last visits to each descendant is greater than the step number of second visit to the gate. Gates 'G4' and 'G5' do not satisfy the first condition as the minimum of first visits to each descendant is smaller than the step number of the first visit to the gates.

The gates 'Top', 'G1', 'G3' and 'G6' can be identified as modules. The top event of the fault tree will always be a module.

Each of the modules can be replaced by modular events in the fault tree structure. Gate 'G1' is replaced by event 'M1', gate 'G3' - by event 'M3' and gate 'G6' is replaced by event 'M2'. Therefore the fault tree shown in Figure 9.1 can be replaced by separate fault trees shown in Figure 9.2.

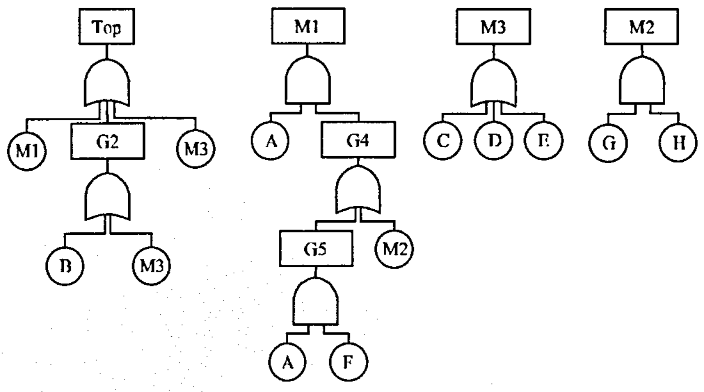


Figure 9.2: Modularised fault tree and modules

9.3 Modularisation for Phased Mission Systems

Cause-consequence diagrams for phased mission systems can be very big and complicated as failure of each component needs to be considered in each phase, not only in the phases that it is used in. Therefore modularisation has been investigated to reduce the size and complexity of the diagrams and ease the analysis.

Two cases of modularisation for phased mission systems were considered:

1. when failures of each phase are important;

2. when only failure of a mission is of interest.

9.3.1 Modularisation of Each Phase of a Phased Mission System

For the first case, when failures of each phase need to be determined, modularisation doesn't offer a great improvement. This is because fault trees for each phase contain basic events that are repeated through out the whole mission and also are inconsistent. For example, the same event 'component A fails in phase 1' might be repeated through all fault trees of the phased mission system.

Consider example shown in Figure 9.3. If Rauzy's algorithm was applied to these fault trees, there would be one module common to both phases (gate 'G1'). The fault trees representing this are shown in Figure 9.4.

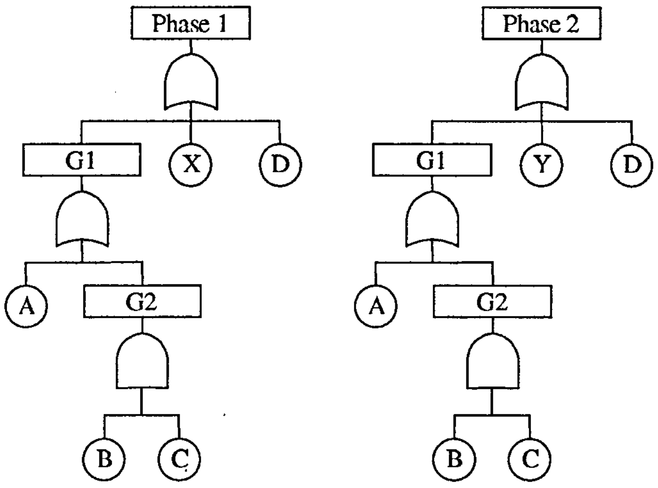


Figure 9.3: Two phase example

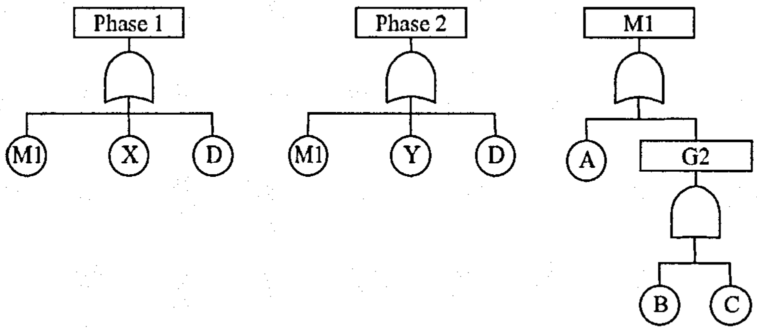


Figure 9.4: Two phase example: fault trees after modularisation

The next step is to apply basic event transformation, so that failures of the components in different phases would be accounted for. Fault trees after basic event transformation are shown in Figure 9.5.

As can be seen from Figure 9.5, although modules 'M1_1' and 'M1_2' are independent for each phase, it is not true for the whole mission. Failure event 'A1' is an input in gate 'M1_1' for Phase 1 and in gate 'M1_2' for Phase 2. The same is true for failure event B1.

Once modules are determined, the cause-consequence diagram can be constructed. The ordering chosen for this example is:

$$M1_1 > X1 > D1 > M1_2 > Y1 > Y2 > D2$$

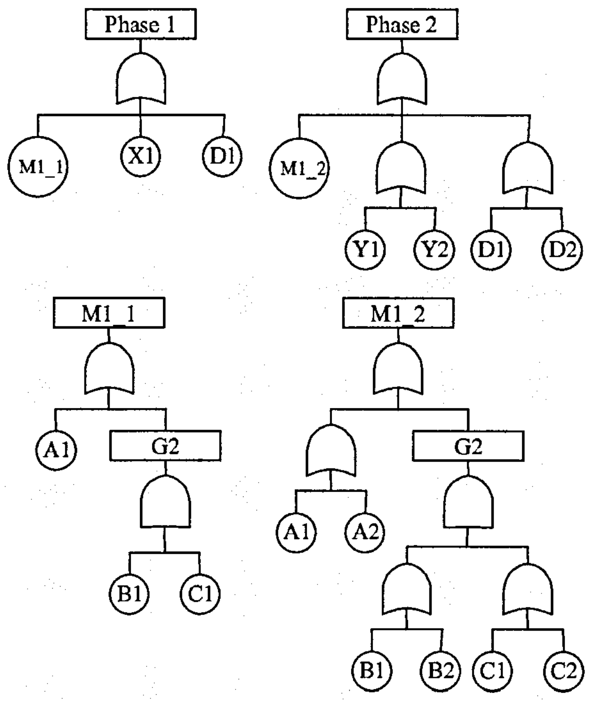


Figure 9.5: Two phase example: fault trees after basic event transformation

The cause-consequence diagram for this phased mission after modularisation is shown in Figure 9.6 with cause-consequence diagrams for the independent modules shown in Figure 9.7.

The quantification of the cause-consequence diagram shown in Figure 9.6 is not straight forward. The quantification technique used for cause-consequence diagrams without modularisation cannot be applied in this case, because there are repeated and inconsistent events throughout the cause-consequence diagram within different modules. For example, basic event 'A1' appears in module 'M1_1' and then again in module 'M1_2'. Also, if failure of component A occurs while the system is in Phase 1, failure of the same component cannot occur in Phase 2, which has basic event event 'A2' (module 'M1_2'). Therefore, to be able to perform quantification on this cause-consequence diagram, it needs to be expanded to its full size and repeated and inconsistent events have to be removed. First step in this would be to

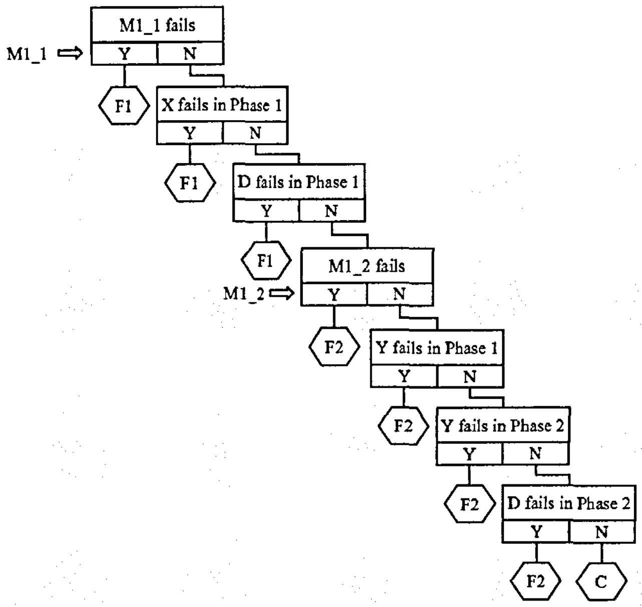


Figure 9.6: Cause-consequence diagram after modularisation

substitute a decision box of each module with the cause-consequence diagram for it. The expanded cause-consequence diagram is shown in Figure 9.8 and Figure 9.9. In these cause-consequence diagrams repeated events are crossed out with solid line. Events crossed out with a dashed line are removed because they are on the redundant branch of the repeated decision box.

After removing irrelevant events the diagram is shown in Figure 9.10. At this point the previously described quantification technique can be used.

The resulting cause-consequence diagram is the same size as if the diagram was constructed using construction Method 1 as described in Section 7.3.1. Therefore, modularisation did not offer any benefit in this example for reducing size or complexity of the resulting cause-consequence diagram. However, if the system contains a subsystem which needs to function only for one phase of the mission, the modularisation could be beneficial.

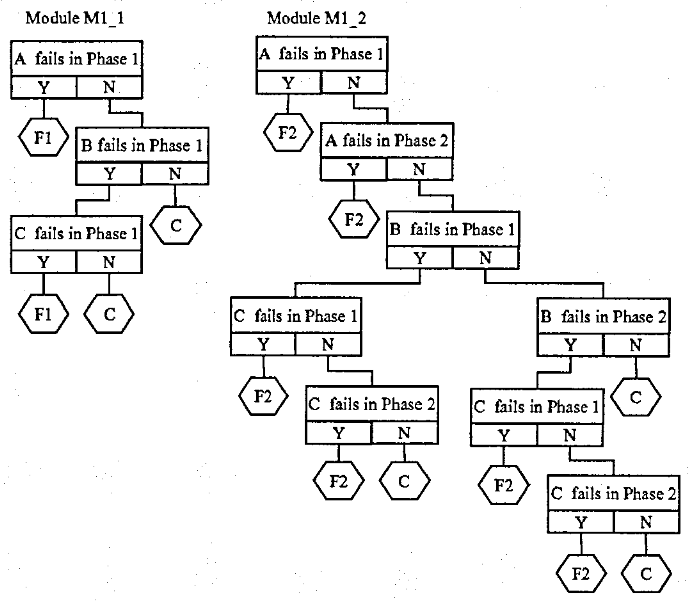


Figure 9.7: Cause-consequence diagrams for modules

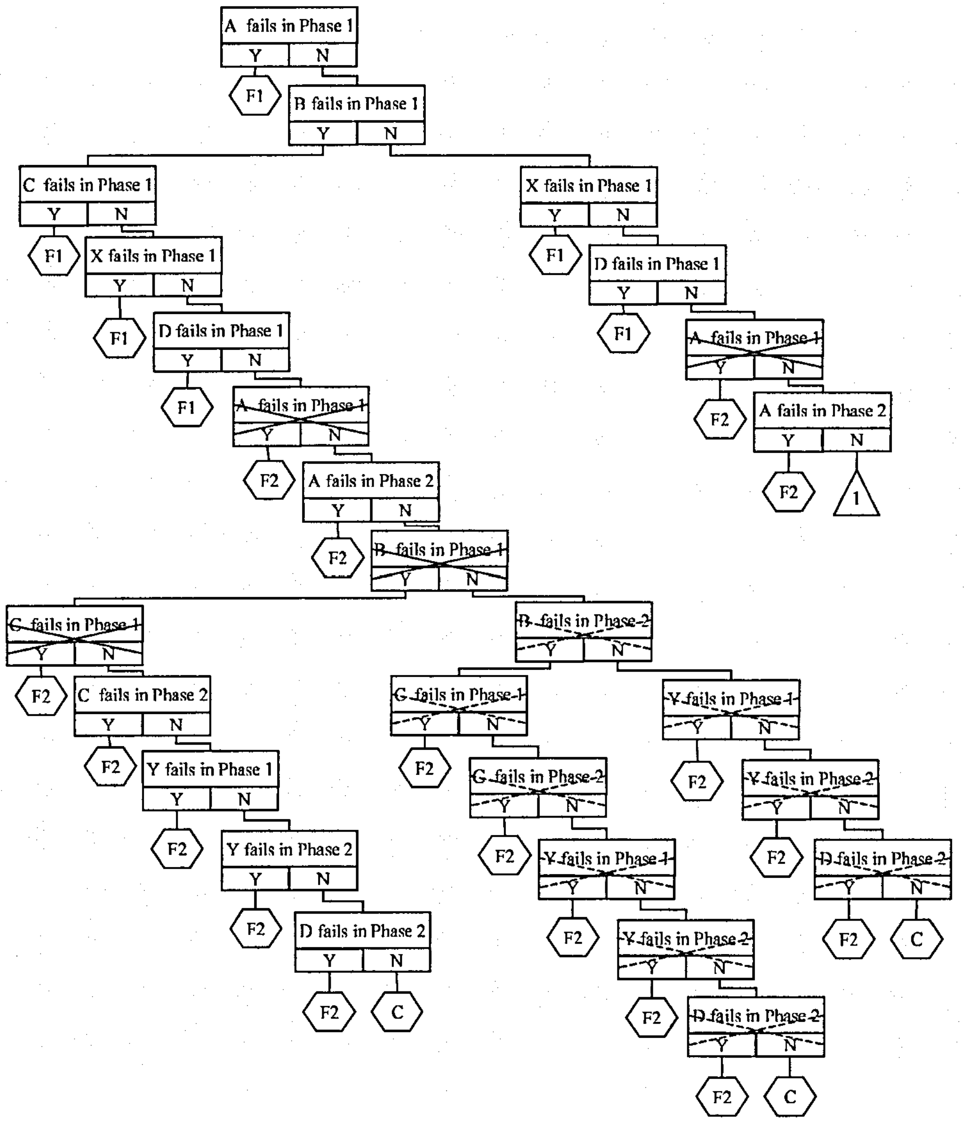


Figure 9.8: Expanding cause-consequence diagram

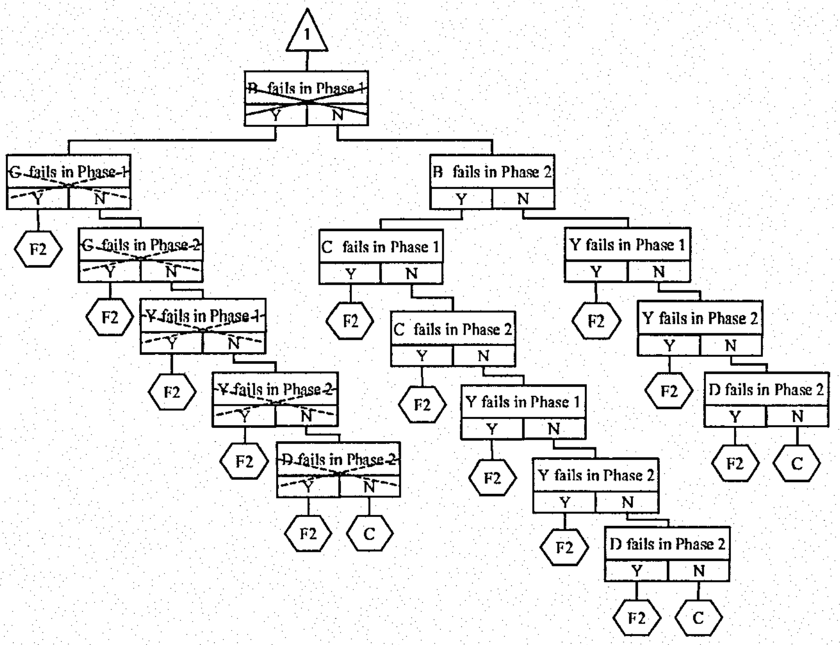


Figure 9.9: (continued) Expanding cause-consequence diagram

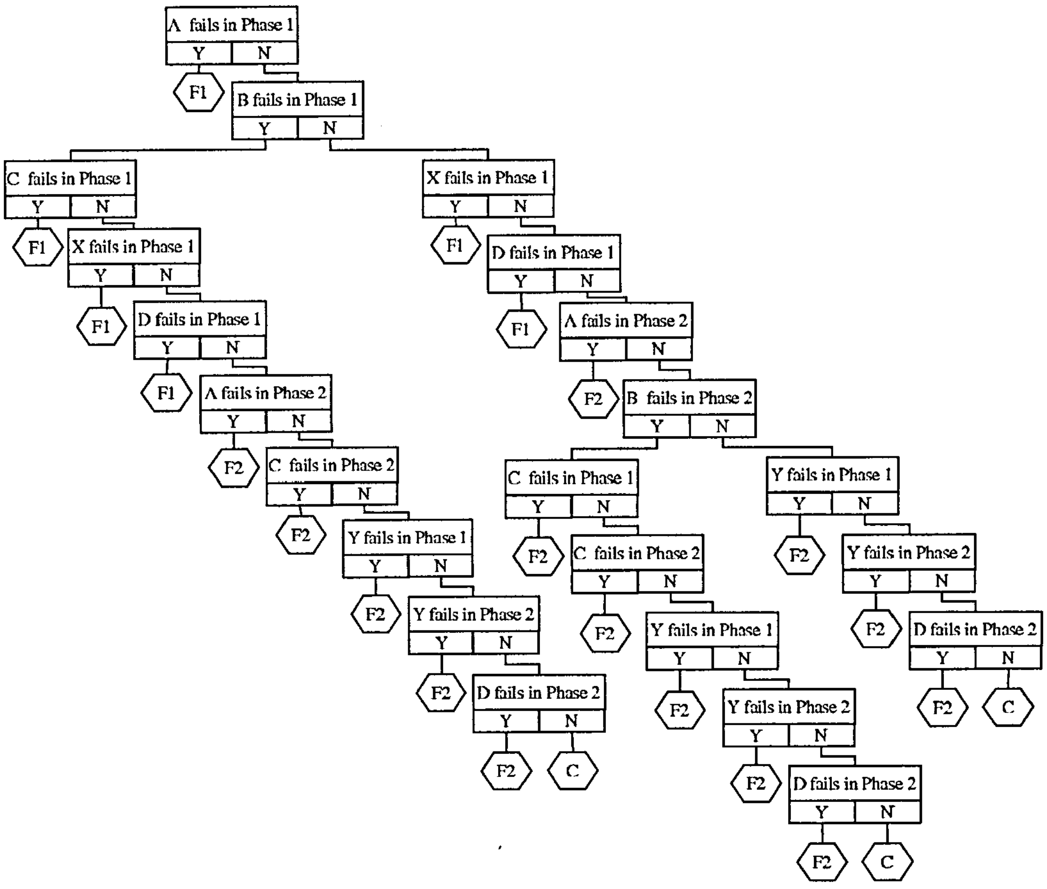


Figure 9.10: Minimized cause-consequence diagram

9.3.2 Modularisation of a Phased Mission System as a Whole

Some phased mission systems may have common modules between the phases which are independent from the rest of of the system at any phase. An example of this could be a specific subsystem that is required to work in all or some of the phases, and which inputs are not used in any other way. In this case, such a subsystem could be quantified separately from the rest of the system, but only if failure or success probability of the phase is not important.

To illustrate this case, the example shown in Figure 9.3 is used. Gate 'G1' is an independent module that appears in both phases, therefore it can be replaced by modular event M1. The modified fault trees are shown in Figure 9.4.

Before constructing the cause-consequence diagram, basic event transformation needs to be performed to take into account component failures in different phases. As failures of separate phases are not investigated, the basic event transformation is not applied to the modular event itself, but to the module. For the module component failures are considered up to the latest phase in which modular event representing the particular module occurs. For example, if it is a 5-phase system and a modular event is occurring in phases 1, 2 and 4, then the basic events in the module would be replaced by an OR combination of failure events for phases 1 to 4. The fault trees after basic event transformation are shown in Figure 9.11.

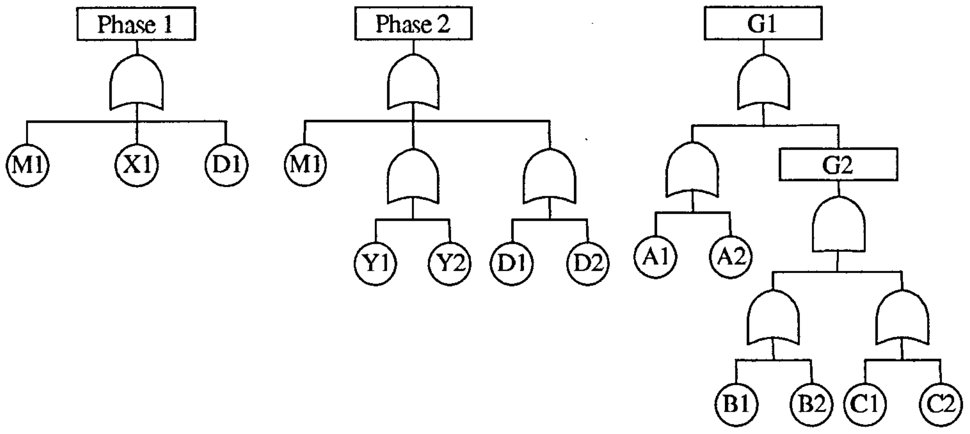


Figure 9.11: Modularised fault trees after basic event transformation

After basic event transformation is applied, the cause-consequence diagram can be constructed. The ordering used for this example is:

$$M1 > X1 > D1 > Y1 > Y2 > D2$$

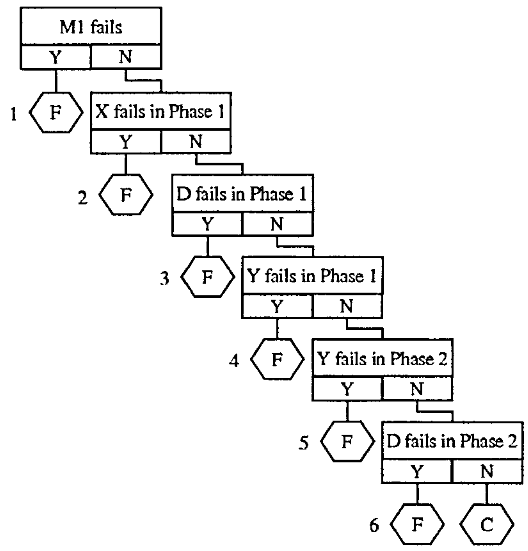


Figure 9.12: Cause-consequence diagram after modularisation

The cause consequence diagram for the phased mission when the phase failures are not investigated is shown in Figure 9.12. The quantification can be performed on this cause-consequence diagram using the same procedure as for cause-consequence diagram without modularisation. The cause-consequence diagram can also be constructed for the module and is shown in Figure 9.13.

9.3.2.1 Qualitative and Quantitative Analysis

Once the cause-consequence diagram is constructed, qualitative and quantitative analysis can be performed. In the case when failures of the phases are not considered, there are two possible outcomes: mission failure (F) or mission success (C).

The qualitative analysis of the cause-consequence diagram will produce the list of causes for mission success or failure. Conditions causing an outcome event to occur are established by investigating each decision box on the path to the outcome and listing the component failure or success as indicated by the exit path from the decision box. The failure conditions for the system shown in Figure 9.3 are listed below:

1. $M1$

2. $\overline{M1} \wedge X1$

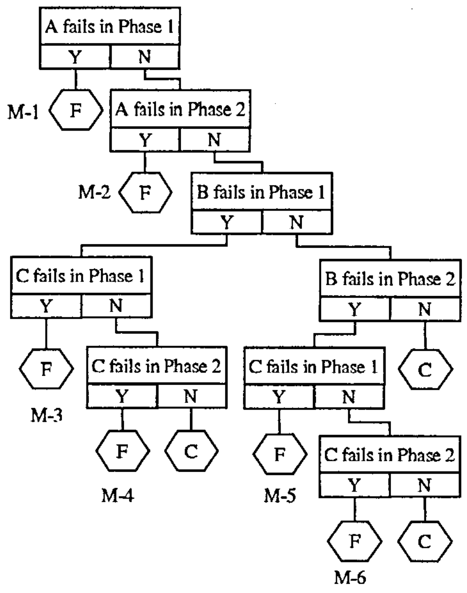


Figure 9.13: Cause-consequence diagram for module

3. $\overline{M1} \wedge \overline{X1} \wedge D1$

4. $\overline{M1} \wedge \overline{X1} \wedge \overline{D1} \wedge Y1$

5. $\overline{M1} \wedge \overline{X1} \wedge \overline{D1} \wedge \overline{Y1} \wedge Y2$

6. $\overline{M1} \wedge \overline{X1} \wedge \overline{D1} \wedge \overline{Y1} \wedge \overline{Y2} \wedge D2$

This list can be simplified as there are some outcomes, that contain a progression of states of the same component. For example, outcome 5 has component Y working in phase 1 and then failing in phase 2. In this case, the event that component Y is working in phase 1 can be removed as its failure in the later phase implies that it was working before. The simplified list of mission failures is listed below:

1. $M1$

2. $\overline{M1} \wedge X1$

3. $\overline{M1} \wedge \overline{X1} \wedge D1$

4. $\overline{M1} \wedge \overline{X1} \wedge \overline{D1} \wedge Y1$

5. $\overline{M1} \wedge \overline{X1} \wedge \overline{D1} \wedge Y2$

6. $\overline{M1} \wedge \overline{X1} \wedge \overline{Y1} \wedge \overline{Y2} \wedge D2$

The reduced, or simplified, list of the component conditions leading to each outcome are in an appropriate form for quantification. The probabilities of each outcome in the list are listed below:

1. $P(M1)$
2. $(1 - P(M1)) \cdot X1$
3. $(1 - P(M1)) \cdot (1 - P(X1)) \cdot D1$
4. $(1 - P(M1)) \cdot (1 - P(X1)) \cdot (1 - P(D1)) \cdot Y1$
5. $(1 - P(M1)) \cdot (1 - P(X1)) \cdot (1 - P(D1)) \cdot Y2$
6. $(1 - P(M1)) \cdot (1 - P(X1)) \cdot (1 - P(Y1) - P(Y2)) \cdot D2$

The sum of all these probabilities would give the exact probability for mission failure.

Probability for the modular event 'M1' is not known and needs to be obtained using fault tree analysis or other reliability analysis tool. It can be obtained using cause-consequence diagram and this will be shown below. The cause-consequence diagram for modular event is shown in Figure 9.13.

Following the same rules as before, the list of component condition for each outcome are listed below:

M-1. $A1$

M-2. $\overline{A1} \wedge A2$

M-3. $\overline{A1} \wedge \overline{A2} \wedge B1 \wedge C1$

M-4. $\overline{A1} \wedge \overline{A2} \wedge B1 \wedge \overline{C1} \wedge C2$

M-5. $\overline{A1} \wedge \overline{A2} \wedge \overline{B1} \wedge B2 \wedge C1$

M-6. $\overline{A1} \wedge \overline{A2} \wedge \overline{B1} \wedge B2 \wedge \overline{C1} \wedge C2$

The reduced (simplified) list would be as follows:

M-1. $A1$

M-2. $A2$

M-3. $\overline{A1} \wedge \overline{A2} \wedge B1 \wedge C1$

M-4. $\overline{A1} \wedge \overline{A2} \wedge B1 \wedge C2$

M-5. $\overline{A1} \wedge \overline{A2} \wedge B2 \wedge C1$

M-6. $\overline{A1} \wedge \overline{A2} \wedge B2 \wedge C2$

Probabilities for all the outcomes of the cause-consequence diagram of modular event are listed below and the probability of the failure of modular event would be obtained by summing the probabilities of these outcome events.

M-1. $P(A1)$

M-2. $P(A2)$

M-3. $(1 - P(A1) - P(A2)) \cdot P(B1) \cdot P(C1)$

M-4. $(1 - P(A1) - P(A2)) \cdot P(B1) \cdot P(C2)$

M-5. $(1 - P(A1) - P(A2)) \cdot P(B2) \cdot P(C1)$

M-6. $(1 - P(A1) - P(A2)) \cdot P(B2) \cdot P(C2)$

9.4 Discussion

If modularisation is applied to fault trees before the construction of the cause-consequence diagram, there are two different cases to be considered. If the failures in phases need to be investigated, then modularisation doesn't reduce the size of the final cause-consequence diagram significantly, as modules in different phases need to be substituted back into the main diagram before quantification. The time consumed could increase as first the modules in the fault trees need to identified, then the cause-consequence diagram constructed and modules need to be substituted back in the cause-consequence diagram before performing qualitative or quantitative assessment.

Another case is if the failures of each phase are not being investigated. In this case modularisation can help to reduce the size of the final cause-consequence diagram produced and separate modules can be quantified separately. This could be especially useful when there are subsystems in the mission that work independently from the rest.

10. CONCLUSIONS AND FUTURE WORK

10.1 Conclusions

The aim of this research was to develop a method for phased mission analysis using cause-consequence diagrams. Cause-consequence diagram method has been successfully implemented for the analysis of phased mission systems.

This method is superior over fault tree analysis as:

- It can represent a whole system in one diagram without missing out valuable information about phase failures.
- The cause-consequence diagram method provides an easy and efficient way to perform quantitative and qualitative analysis on phased missions avoiding approximations that could result in inaccuracies.
- The diagram is easy to follow and contains descriptions of components states and as such can be presented to those without much prior knowledge in risk analysis.

In addition to these conclusions, the cause-consequence diagram method is superior over binary decision diagrams as:

- It can represent a whole system in one diagram without missing out valuable information about phase failures. The binary decision diagrams can be used to obtain mission failure probability, but if the phase failure probabilities are of interest, separate binary decision diagrams would have to be constructed for each phase. When using cause-consequence diagram method, phase failure probability is calculated in addition to mission failure probability.
- The diagram is easy to follow and contains descriptions of components states. Therefore it can be presented to those without much prior knowledge in risk analysis.

The size of the diagram is greatly dependent on the ordering of the components, just like with the binary decision diagrams, and the selection of an effective variable ordering scheme can influence the efficiency of the cause-consequence diagram for the phased mission problem and therefore the computational resources necessary to calculate the result. As has been shown in this work for the same phased mission system, the cause-consequence diagram method is more efficient than a binary decision diagram in terms of size.

Modularisation is widely used in fault tree analysis, where it effectively reduces the size of the problem. The advantage of this technique is that each module can be regarded as an individual fault tree and analysed independently. The same principle was applied to cause-consequence diagrams to investigate its effectiveness in this context.

- When phase failures are of interest, modularisation does not reduce the size of the final cause-consequence diagram significantly, because modules in different phases need to be substituted back into the main diagram before quantification. The time consumed could even increase because initially the modules in the fault trees would have to be identified, then the corresponding cause-consequence diagrams constructed for the modules and the main fault tree, and finally those modules need to be substituted back in the main cause-consequence diagram before performing qualitative or quantitative assessment.

- The alternative case is when the failure of each phase is not being investigated, as only the overall mission success or failure is of interest. In this situation, modularisation can help to reduce the size of the final cause-consequence diagram produced, because the separate modules can be quantified individually. This advantage is particularly evident when there are subsystems in the mission working independently from the rest.

The work presented in this thesis has been applied to an aircraft flight. The system was simplified for the scope of this thesis. The results obtained could not be compared with the real life results as no real data was available for the system and also because of the simplified approach to the example. Exact and coherent approximate quantification techniques were used on the cause-consequence diagram, both of them producing very close results.

10.2 Recommendations for future work

The current methods for construction CCD are based on the principle of converting fault trees into CCD. This process is not very intuitive and requires the use of trained personnel. If CCD could be constructed directly from the description of the system, a significant saving could be achieved as a result of omitting the construction of system fault trees.

The component ordering schemes for single-phased mission binary decision diagram have been widely researched. As logics of cause-consequence diagrams and binary decision diagrams are very similar, the same ordering principle can be applied for both. Therefore, it would be useful to be able to obtain an optimal event ordering scheme for phased mission system that would result in the most efficient cause-consequence diagram for phased mission.

The work could be extended to include phased missions with one or more repairable states. This may involve combining different methods available for phased mission analysis, such as Markov analysis and simulation techniques.

The example of an aircraft flight presented in the thesis was simplified and only main sub-systems were included. This could be reviewed to expand the sub-systems to the following levels and to represent a more realistic system to test the approach. Also, as the data used for the calculations was randomly generated, it would be useful to obtain the data from industry for the component failures and to compare the results obtained using cause-consequence diagram with the real life situation. Also, more real-life examples should be used to illustrate the benefits of the method, such as applications in automotive industry, space applications and military operations (ballistic missile).

As only method 1 was coded for this work, it would be useful to provide a code for method 2, so that the methods proposed in the thesis could be compared for their efficiency.

BIBLIOGRAPHY

[1] Burdick GR, Fussell JB, Rasmuson DM, Wilson JR. *Phased Mission Analysis: A Review of New Developments and An Application.* IEEE Transaction on Reliability, Vol.R-26, No.1, 1997, pp 43-49

[2] Esary JD, Ziehms H. *Reliability analysis of phased missions.* Reliability and Fault Tree Analysis: Theoretical and Applied Aspects of system, Reliability and Safety Assessment, Philadelphia, 1975, pp 213-236

[3] La Band R.A., Andrews J.D., *Phased Mission Modelling Using Fault Tree Analysis*, Proc Instn Mech Engrs Part E: Journal of Process Mechanical Engineering, 218[2] 83-91 (2004)

[4] Zang X, Sun H, Trivedi KS. *A BDD-Based Algorithm for Reliability Analysis of Phased-Mission Systems.* IEEE Transactions on Reliability, Vol.48, No.1, 1999, pp 50-60

[5] Nielsen DS. *The Cause/Consequence Diagram Method as a Basis for Quantitative Accident analysis.* Danish Atomic Energy Commission, 1971 RISO-M-1374

[6] Andrews JD, Ridley LM. *Reliability of sequential systems using the cause-consequence diagram method.* IMechEProceedings Part E, 2001, pp 207-220

[7] Dhillon BS, Singh C. *Engineering Reliability: New Techniques and Applications.* New York , Chichester, Wiley, 1981

[8] Vesely, WE. *A Time Dependent Methodology for Fault Tree Evaluation.* Nuclear Design and engineering, 13, pp337-360, 1970

[9] Lee CY. *Representation of Switching Circuits by Binary-Decision Programs.* The Bell Technical Journal, 38, pp985-999, July 1959

[10] Akers SB. *Binary Decision Diagrams.* EEE Transactions on Computers, 27, No.6, pp509-516, June 1978

[11] Schneeweiss WG. *Fault-Tree Analysis Using a Binary Decision Tree.* IEEE Transactions on Reliability, 34, No.5, pp453-457, 1985

[12] Rauzy A. *New algorithms for fault trees analysis.* Reliability Engineering and system Safety, 40, pp203-211, 1993

[13] Friedman SJ, Supowit KJ. *inding the Optimal Variable Ordering for Binary Decision Diagrams.* IEEE Transactions on Computers, 39, No.5, pp710-713, May 1990

[14] Nielsen DS, Runge B. *Unreliability of a Standby System with Repair and Imperfect Switching.* IEEE Transactions on Reliability, 1974, Vol.R-23, pp 17-24

[15] Nielsen DS. *Use of Cause-Consequence Charts in Practical Systems Analysis.* Reliability and Fault Tree Analysis: Theoretical and Applied Aspects of system, Reliability and Safety Assessment, Philadelphia, 1975, pp 849-880

[16] Burdick GR, Fussell JB. *On the Adaptation of Cause-Consequence Analysis to U.S. Nuclear Power systems Reliability and Risk Assessment.* A Collection of Methods for Reliability and Safety Engineering, Report V, Idaho National Engineering Laboratory, 1976, ANCR-1273

[17] Villemeur A. *Reliability, Availability, Maintainability and Safety Assessment.* Volume 1, Willey, Chichester, 1991

[18] Hickling P. *The use of cause-consequence diagrams for the reliability analysis of sequentially operating systems.* British Gas Report, 1980

[19] Nielsen DS, Platz O, Runge B. *A Cause-Consequence Chart of a Redundant Protection System.* IEEE Transactions on Reliability, 1975 Vol.R-24, No.1, pp 8-13

[20] Andrews JD, Moss TR. *Reliability and Risk Assessment.* Professional Engineering Pub., London, 2002

[21] Evans RA. *Fault-Trees and Cause-Consequence Charts.* IEEE Transactions on Reliability, 1974, Vol.R-23, No.1, pp 1

[22] Ridley LM. *Dependency Modelling Using Fault Tree and Cause-Consequence Diagram.* Thesis, Loughborough University, 2000

[23] Andrews JD, Ridley LM. *Application of cause-consequence diagram method to static systems.* Reliability Engineering and System Safety, 2002, Vol.75. pp 47-58

[24] Nielsen DS, Platz O, Kongso HE. *Reliability Analysis of Proposed Instrument Air System.* Danish Atomic Energy Commission, 1977, RISO-M-1903

[25] Taylor JR. *Interlock Design Using Fault Tree and Cause Consequence Analysis.* Danish Atomic Energy Commission, 1976, RISO-M-1890

[26] Dutuit Y, Rauzy A. *A Linear-Time Algorithm to Find Modules of Fault Tree.* IEEE Transactions on Reliability, 45, No.3, 1996

[27] Burdick GR, Fussell JB, Rasmuson DM, Wilson JR. *The Implementation of Phased Mission Techniques To Nuclear Systems Analysis.* A Collection of Methods for Reliability and Safety Engineering, Report V, Idaho National Engineering Laboratory, 1976, ANCR-1273

[28] Ma Y, Trivedi K S *An Algorithm for Reliability Analysis of Phased-Mission Systems*, Reliability Engineering and Systems Safety 66 (1999) 157-170

[29] Somani AK, Trivedi KS. *Phased-Mission System Analysis Using Boolean Algebraic Methods.* ACM SYETRICS Performance Evaluation Review, Vol.22, Issue 1, 1994, pp 98-107

[30] Montague DF, Fussell JB. *A Methodology for Calculating the expected Number of Failures of a System Undergoing a Phased Mission.* Nuclear Science and engineering, Vol.74, 1980, pp 199-209

[31] Pedar A, Sarma VVS. *Phased-Mission Analysis for Evaluating the Effectiveness of Aerospace Computing-Systems.* IEEE Transactions on Reliability, Vol.R-30, No.5, 1981, pp 429-436

[32] Langberg N, Proschan F, Quinzi AJ. *Converting Dependent Models into Independent Ones, with Applications in Reliability.* The Theory and Applications of Reliability, Volume 1, Academic Press, London, New York, 1977

[33] Veatch MH. *Reliability of Periodic, Coherent, Binary Systems.* IEEE Transactions on Reliability, Vol.R-35, No.5, 1986, pp 504-507

[34] Dazhi X, Xiaozhong W. *A Practical Approach for Phased Mission Analysis.* Reliability Engineering and System Safety, Vol.25, 1989, pp 333-347

[35] Lee KW, Hong JS. *Reliability Evaluation of a Phased mission System with Time Varying Redundancy and Failure Probability.* Microelectronics and Reliability, Vol.31, No.5, 1991, pp 955-961

[36] Xing L, Dugan JB. *Analysis of Generalized Phased-Mission System Reliability, Performance, and Sensitivity.* IEEE Transactions on Reliability, Vol.51, No.2, 2002, pp 199-211

[37] Clarotti CA, Contini S, Somma R. *Repairable Multiphase Systems - Markov and Fault-Tree Approaches for Reliability Evaluation.* Synthesis and Analysis methods for Safety and Reliability Studies, 1980, pp 45-58

[38] Alam M, Al-Saggaf UM. *Quantitative Reliability Evaluation of Repairable Phased-Mission Systems Using Markov Approach.* IEEE Transactions on Reliability, Vol.R-35, No.5, 1986, pp 498-503

[39] Smotherman MK, Geist RM. *Phased Mission effectiveness using a Nonhomogeneous Markov Reward Model.* Reliability and System Safety, Vol.27, 1990, pp 241-255

[40] Smotherman M, Zemoudeh K. *A Non-Homogeneous Markov Model for Phased-Mission Reliability Analysis.* IEEE Transactions on Reliability, Vol.38, No.5, 1989, pp 585-590

[41] Dugan JB. *Automated analysis of Phased-Mission Reliability.* IEEE Transactions on Reliability, Vol.40, No.1, 1991, pp 45-52