Fault Tree Handbook with Aerospace Applications

Prepared for
NASA Office of Safety and Mission Assurance
NASA Headquarters
Washington, DC 20546

August, 2002

Fault Tree Handbook with Aerospace Applications

NASA Project Coordinators:

Dr. Michael Stamatelatos, NASA Headquarters
Office of Safety and Mission Assurance

Mr. José Caraballo, NASA Langley Research Center

Authors:

NASA

Dr. Michael Stamatelatos, NASA HQ, OSMA

Lead Author:

Dr. William Vesely, SAIC

Contributing Authors (listed in alphabetic order):

Dr. Joanne Dugan, University of Virginia
Mr. Joseph Fragola, SAIC
Mr. Joseph Minarick III, SAIC
Mr. Jan Railsback, NASA JSC

Fault Tree Handbook with Aerospace Applications

Acknowledgements

The project coordinators and the authors express their gratitude to NASA Office of Safety and Mission Assurance (OSMA) management (Dr. Michael Greenfield, Deputy Associate Administrator and Dr. Peter Rutledge, Director of Enterprise Safety and Mission Assurance) and to Mr. Frederick Gregory, NASA Deputy Administrator, for their support and encouragement in developing this document. The authors also owe thanks to a number of reviewers who provided constructive criticism.

Fault Tree Handbook with Aerospace Applications

Foreword

NASA has been a leader in most technologies it has employed in its programs over the years. One of the important NASA objectives is now to add Probabilistic Risk Assessment (PRA) to its repertoire of expertise in proven methods to reduce technological and programmatic risk.

Fault Tree Analysis (FTA) is one of the most important logic and probabilistic techniques used in PRA and system reliability assessment today.

Methods to perform risk and reliability assessment in the early 1960s originated in US aerospace and missile programs. Fault tree analysis is such an example that was quite popular in the mid sixties. Early in the Apollo project the question was asked about the probability of successfully sending astronauts to the moon and returning them safely to Earth. A risk, or reliability, calculation of some sort was performed and the result was a mission success probability that was unacceptably low. This result discouraged NASA from further quantitative risk or reliability analysis until after the Challenger accident in 1986. Instead, NASA decided to rely on the use of failure modes and effects analysis (FMEA) and other qualitative methods for system safety assessments. After the Challenger accident, the importance of PRA and FTA in systems risk and reliability analysis was realized and its use at NASA has begun to grow.

The nuclear industry began to utilize probabilistic risk assessment to assess safety following the Three Mile Island accident in 1979. In 1981, the US Nuclear Regulatory Commission (NRC) issued the Fault Tree Handbook, NUREG-0492. Over the past two decades, this document has become the leading technical information source on how FTA should be performed. Although originally intended for nuclear power applications, the Fault Tree Handbook has been extensively used in all fields where this powerful systems analysis methodology was applied.

Over the past two decades, probabilistic risk assessment and its underlying techniques, including FTA, has become a useful and respected methodology for safety assessment. Because of its logical, systematic and comprehensive approach, PRA and FTA have been repeatedly proven

capable of uncovering design and operational weaknesses that escaped even some of the best deterministic safety and engineering experts. This methodology showed that it was very important to examine not only low-probability and high-consequence individual mishap events, but also high-consequence scenarios which can emerge as a result of occurrence of multiple high-probability and nearly benign events. Contrary to common perception, the latter is oftentimes more detrimental to safety than the former.

A foremost strength of PRA and its underlying analysis techniques, including FTA, is that it is a decision support tool. In safety applications, this methodology helps managers and engineers find design and operational weaknesses in complex systems and then helps them systematically and efficiently uncover and prioritize safety improvements.

In order to best benefit from PRA and FTA in management decisions, it is important that managers and their support staffs be familiar with the value and application of these methods. In addition, there should be a small but robust group of in-house technical experts that understand the methods used in a PRA or FTA study, can explain its meaning and applicability to given problems to management and serve as in-house technical advisers to the management decision process for safety improvement. If these in-house experts do not exist initially, they should be hired or groomed through training and transfer of technology, becoming part of the corporate resources and memory that will help shape the organization, taking advantage of the PRA and FTA methods and results and the expert knowledge of the external consultants. In-house experts will help build risk-based knowledge and experience and stimulate cultural changes so that a progressive organization can use these resources to make sound and cost-effective safety improvement decisions.

In support of this, NASA has recently began to implement the following important risk assessment enhancement principles in its programs and projects:

- Transfer quantitative risk assessment technology to NASA managers and practitioners as soon as possible,
- Develop or acquire quantitative risk assessment expertise and state-of-the-art software and data,
- Encourage ownership in quantitative risk assessment methods, studies and results in order to use them effectively in the management decision process,
- Develop a corporate memory of the risk assessment project results and data on which to build future capabilities and experience, and
- Develop risk awareness in programs and projects that will eventually help NASA develop a risk-informed culture for all its programs and activities.

To this end, and in support of the Risk Management Program, NASA began to develop training and practitioner documents on how to perform quantitative risk assessment and utilize important techniques like FTA. One such document is a Procedures Guide for performing PRA for aerospace applications. The other is this document, the re-issue of an updated version of the Fault Tree Handbook for aerospace applications.

A considerable amount of material on PRA methods and applications has been written over the past three decades. Several university and practitioner textbooks and sourcebooks currently exist

but they focus on application of PRA in industries other than aerospace. Although some of the techniques used in PRA originated in work for aerospace and military applications, no comprehensive reference currently exists for PRA applications to aerospace systems. In particular, no comprehensive reference for applying FTA to aerospace systems currently exists.

The current Fault Tree Handbook, serves two purposes:

- As a companion document to the training material taught in FTA courses for practicing system analysts, and
- To assist aerospace FTA practitioners in acquiring and implementing current state-of-the art FTA techniques in their applications.

The Handbook contains some of the material of the original handbook. However, some of the basic tutorial material from the original handbook was eliminated because currently, unlike the time when this handbook was first published, a number of PRA textbooks containing this type of material are in existence.

The current version of the Fault Tree Handbook contains the following material that was not in the original version:

- A discussion of the Binary Decision Diagram (BDD) method for solving fault trees that were originally solved only through Boolean reduction and the use of minimal cuts sets;
- An introduction to Dynamic Fault Trees (DFTs) and methods to solve them;
- Illustrations of fault tree analysis in aerospace applications, with detailed description of the models;
- An extended discussion of modeling common cause failures and human errors in FTA;
- Descriptions of modeling feedback loops so as to properly cut such loops in a FT;
- Extended discussion of applications of FTA for decision making, covering applications to operating systems and to systems that are in design;
- Descriptions of absolute and relative importance measures that are obtainable from FTA and that enhance the output and value of an FTA; and
- Expanded discussion of success trees, their logical equivalence to fault trees, and their applications.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

1. Introduction and Overview

1.1 Introduction and Intended Readers

This handbook is an update of the original Fault Tree Handbook published in 1981 [1]. It is written for the informed reader who has some knowledge of system analysis and has knowledge of basic mathematics. This handbook is intended for system analysts, system engineers, and managers. No previous knowledge or training in statistics, reliability, or risk analysis is assumed. Basic concepts of statistical analysis, reliability analysis, and risk analysis are presented in relevant chapters and in the appendices.

This updated version of the Fault Tree Handbook is entitled *Fault Tree Handbook with Aerospace Applications* or AFTH for short. The AFTH presents the basic principles and procedures for Fault Tree Analysis (FTA), with an emphasis on Aerospace applications. The AFTH is organized into two major parts.

The first part of the handbook describes the concepts, steps, tools, and uses of FTA. FTA is a deductive, failure-based approach. As a deductive approach, FTA starts with an undesired event, such as failure of a main engine, and then determines (deduces) its causes using a systematic, backward-stepping process. In determining the causes, a fault tree (FT) is constructed as a logical illustration of the events and their relationships that are necessary and sufficient to result in the undesired event, or top event. The symbols used in a FT indicate the type of events and type of relationships that are involved. The FT is a qualitative model that provides extremely useful information on the causes of the undesired event. The FT can also be quantified to provide useful information on the probability of the top event occurring and the importance of all the causes and events modeled in the FT. This handbook leads the reader through FTA. Particular details can be skipped if the reader desires only an overview of FTA and instead wants to focus on its uses to assist decision-making.

In addition to FTA, inductive approaches are also used in safety analysis and in risk and reliability analysis. In contrast to the deductive approach used in FTA, inductive approaches are forward-stepping approaches that begin with a basic cause or initiating event and then investigate (induce) the end effects. Both FTA and inductive approaches are failure-based. The advantages of failure-based approaches are also discussed.

A FT can be transformed into its logical complement, a success tree (ST) that shows the specific ways the undesired event can be prevented from occurring. The ST provides conditions that, if assured, guarantee that the undesired event will not occur. The ST is a valuable tool that provides equivalent information to the fault tree, but from a success viewpoint. Techniques for transforming the FT to its ST are described along with uses of the ST.

The uses of FTA to assist decision-making are described in this AFTH. FTA provides critical information that can be used to prioritize the importance of the contributors to the undesired event. The contributor importances provided by FTA vividly show the causes that are dominant and that should be the focus of any safety or reliability activity. More formal risk-benefit approaches can also be used to optimally allocate resources to minimize both resource expenditures and the occurrence probability of the undesired event. These risk benefit

approaches are useful for allocating resource expenditures, such as safety upgrades to complex systems like the Space Shuttle.

FTA can be applied to both an existing system and to a system that is being designed. When it is applied to a system being designed for which specific data do not exist, FTA can provide an estimate of the failure probability and the important contributors using generic data to bracket the design components or concepts. FTA can also be used as an important element in the development of a performance-based design. When applied to an existing system, FTA can be used to identify weaknesses and to evaluate possible upgrades. It can also be used to monitor and predict behavior. Furthermore, FTA can be used to diagnose causes and potential corrective measures for an observed system failure. The approaches and tools to obtain this information and the applications of this information in decision-making are important topics of the AFTH.

The second part of the AFTH contains examples of the application of FTA in studies that have been previously performed. The focus is on aerospace applications. The examples include the rupture of a pressure tank (a classic FTA example), failure to initiate and terminate thrust in a monopropellant propulsion system, failure of a redundant container seal (design analysis), and a dynamic FT analysis of a mission avionics system.

1.2 The Fault Tree Approach

FTA can be simply described as an analytical technique, whereby an undesired state of the system is specified (usually a state that is critical from a safety or reliability standpoint), and the system is then analyzed in the context of its environment and operation to find all realistic ways in which the undesired event (top event) can occur. The fault tree itself is a graphic model of the various parallel and sequential combinations of faults that will result in the occurrence of the predefined undesired event. The faults can be events that are associated with component hardware failures, human errors, software errors, or any other pertinent events which can lead to the undesired event. A fault tree thus depicts the logical interrelationships of basic events that lead to the undesired event, the top event of the fault tree.

It is important to understand that a fault tree is not a model of all possible system failures or all possible causes for system failure. A fault tree is tailored to its top event that corresponds to some particular system failure mode, and the fault tree thus includes only those faults that contribute to this top event. Moreover, these faults are not exhaustive—they cover only the faults that are assessed to be realistic by the analyst.

It is also important to point out that a fault tree is not in itself a quantitative model. It is a qualitative model that can be evaluated quantitatively and often is. This qualitative aspect, of course, is true of virtually all varieties of system models. The fact that a fault tree is a particularly convenient model to quantify does not change the qualitative nature of the model itself.

Intrinsic to a fault tree is the concept that an outcome is a binary event i.e., to either success or failure. A fault tree is composed of a complex of entities known as “gates” that serve to permit or inhibit the passage of fault logic up the tree. The gates show the relationships of events needed for the occurrence of a “higher” event. The “higher” event is the output of the gate; the

"lower" events are the "inputs" to the gate. The gate symbol denotes the type of relationship of the input events required for the output event. Figure 1-1 shows a simple fault tree.

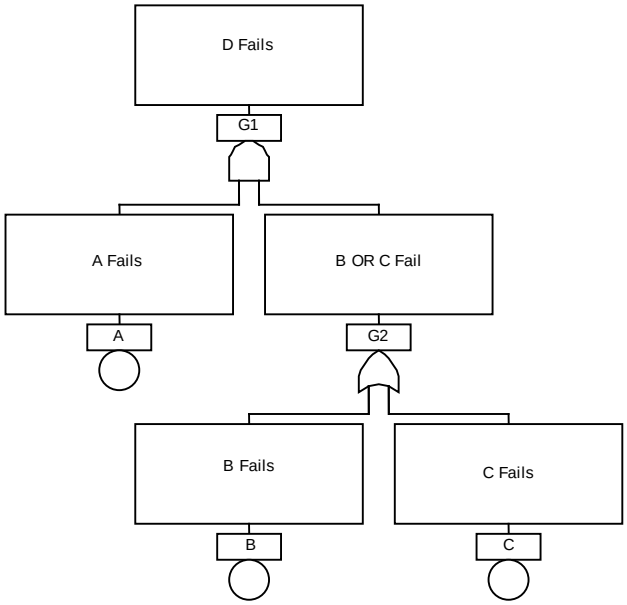


Figure 1-1. A Simplified Fault Tree

1.3 Qualitative and Quantitative Evaluations of a Fault Tree

Both qualitative and quantitative evaluations can be performed on an FT. The FT itself is a qualitative assessment of the events and relationships that lead to the top event. In constructing the FT, significant insights and understanding are gained concerning the causes of the top event. Additional evaluations serve to further refine the information that the FT provides.

The qualitative evaluations basically transform the FT logic into logically equivalent forms that provide more focused information. The principal qualitative results that are obtained are the minimal cut sets (MCSs) of the top event. A cut set is a combination of basic events that can cause the top event. An MCS is the smallest combination of basic events that result in the top event. The basic events are the bottom events of the fault tree. Hence, the minimal cut sets relate the top event directly to the basic event causes. The set of MCSs for the top event represent all the ways that the basic events can cause the top event. A more descriptive name for a minimal cut set may be "minimal failure set." The set of MCSs can not only be obtained for the top event, but for any of the intermediate events (e.g., gate events) in the FT.

A significant amount of information can be obtained from the structure of MCSs. Any MCS with one basic event identifies a single failure or single event that alone can cause the top event to occur. These single failures are often weak links and are the focus of upgrade and prevention actions. Examples of such single failures are a single human error or single component failure that can cause a system failure. An MCS having events with identical characteristics indicates a susceptibility to implicit dependent failure, or common cause, that can negate a redundancy. An example is an MCS of failures of identical valves. A single manufacturing defect or single environmental sensitivity can cause all the valves to simultaneously fail.

The quantitative evaluations of a FT consist of the determination of top event probabilities and basic event importances. Uncertainties in any quantified result can also be determined. Fault trees are typically quantified by calculating the probability of each minimal cut set and by summing all the cut set probabilities. The cut sets are then sorted by probability. The cut sets that contribute significantly to the top event probability are called the dominant cut sets. While the probability of the top event is a primary focus in the analysis, the probability of any intermediate event in the fault tree can also be determined. Different types of probabilities can be calculated for different applications. In addition to a constant probability value that is typically calculated, time-related probabilities can be calculated providing the probability distribution of the time of first occurrence of the top event. Top event frequencies, failure or occurrence rates, and availabilities can also be calculated. These characteristics are particularly applicable if the top event is a system failure.

In addition to the identification of dominant cut sets, importances of the events in the FT are some of the most useful information that can be obtained from FT quantification. Quantified importances allow actions and resources to be prioritized according to the importances of the events causing the top event. The importance of the basic events, the intermediate events, and the minimal cut sets can be determined. Different importance measures can be calculated for different applications. One measure is the contribution of each event to the top event probability. Another is the decrease in the top event probability if the event were prevented from occurring. A third measure is the increase in the top event probability if the event were assured to occur. These importance measures are used in prioritization, prevention activities, upgrade activities, and in maintenance and repair activities. Later sections describe in further detail the rich amount of qualitative and quantitative information that can be obtained from a FT.

1.4 The Success Tree as a Logical Complement of the Fault Tree

Since success and failure are related, the FT can be transformed into its equivalent ST. In the FT context, success in a success tree is specifically defined as the top event not occurring. The method by which the ST can be obtained from the FT will be described in a later section. The ST is a logical complement of the FT, with the top event of the ST being the complement of the top event of the FT. For example, if the top event of the FT is “Occurrence of LOV,” LOV implying Loss of Vehicle, then the ST will have as a top event “Nonoccurrence of LOV.” The ST therefore defines the logic for the failure top event not occurring. Moreover, the ST identifies the minimal sets of basic events that need to be prevented in order to assure that the failure top event will not occur. These minimal sets of events that prevent the failure top event are termed the minimal path sets. A more descriptive name may be “minimal prevention sets” since they indicate how to prevent the occurrence of the failure top event and achieve success in terms of its nonoccurrence. The minimal path sets provide valuable information on the means by which the failure top event can be prevented even without quantification. Moreover, the ST can be quantified to provide the probability of success, i.e., nonoccurrence of the failure top event. Additionally, each of the minimal path sets can be quantified to prioritize the most effective methods for prevention (often in terms of cost to ensure prevention). Ability to analyze the top event from both a failure (occurrence) and success (nonoccurrence) standpoint increases the scope of information that can be obtained from these logic trees.

1.5 Role of FTA in Decision Making

A variety of information is provided by FTA to assist decision-making. An overview some of the major uses of FTA is presented here to give the reader an appreciation of the breadth of applications of FTA in decision-making. Note that this section includes some information already provided in previous sections for the benefit of readers who want to focus on the FTA role in decision making.

1. *Use of FTA* to *understand of the logic leading to the top event*. FTA provides a visual, logic model of the basic causes and intermediate events leading to the top event. Typically, fault trees are not limited to a single system, but cross system boundaries. Because of this, they have shown great benefit in identifying system interactions that impact redundancy. The combination of failures and events that propagate through a system are clearly shown. The minimal cut sets can be organized and prioritized according to the number of events involved and their nature. For example, if there are minimal cut sets that contain only one component failure then this shows that single component failures can cause failure of the system. A failure path of only human errors shows that human errors alone can cause system failure. After reading this handbook, the reader should be convinced that the qualitative information obtained from an FTA is of equal importance to the quantitative information provided.

2. *Use of FTA to prioritize the contributors leading to the top event*. One of the most important types of information from FTA is the prioritization of the contributors to the top event. If a FT is quantified, the failures and basic events that are the causes of the top events can be prioritized according to their importance. In addition, the intermediate faults and events leading to the top event can also be prioritized. Different prioritizations and different importance measures are produced for different applications. One of the valuable conclusions from FTAs is that generally only a few contributors are important to the top event. Often only 10% to 20% of the basic events contribute significantly to the top event probability. Moreover, the contributors often cluster in distinct groupings whose importances differ by orders of magnitude.

   The prioritizations obtained from FTA can provide an important basis for prioritizing resources and costs. Significant reductions in resource expenditures can be achieved with no impact to the system failure probability. For a given resource expenditure, the system failure probability can be minimized by allocating resources to be consistent with contributor importances. The importance measures obtained from a FTA are as important as the top event probability or the ranked cut set lists obtained from the analysis.

3. *Use of FTA as a proactive tool to prevent the top event*. FTA is often used to identify vulnerable areas in a system. These vulnerable areas can be corrected or improved before the top event occurs. Upgrades to the system can be objectively evaluated for their benefits in reducing the probability of the top event. The evaluation of upgrades is an important use of the FTA. Advocates of different corrective measures and

upgrades will often claim that what they are proposing provides the most benefit and they may be correct from their local perspective. However, FTA is a unique tool that provides a global perspective through a systematic and objective measure of the impact of a benefit on the top event. The probability of the top event can be used to determine the criticality of carrying out the upgrades. The probability of the top event can be compared to acceptability criteria or can be used in cost benefit evaluations. Advances in cost benefit methodology allow uncertainties and risk aversion to be incorporated as well as the probabilities. Furthermore, success paths provided from FTA can be used to identify specific measures that will prevent the top event. The proactive use of FTA has been shown to be one of its most beneficial uses.

4. *Use of FTA to monitor the performance of the system.* The use of the FT as a monitoring tool is a specific proactive use that has been identified because of its special features. When monitoring performance with regard to the top event, FTA can account for updates in the basic event data as well as for trending and time dependent behaviors, including aging effects. Using systematic updating techniques, the fault tree can be re-evaluated with new information that can include information on defects and near failures. Actions can then be identified to maintain or replace necessary equipment to control the failure probability and risk. This use of FTA as a monitoring tool is common in the nuclear industry.

5. *Use of FTA to minimize and optimize resources.* This particular use of FTA is sometimes overlooked but it is one of the most important uses. Through its various importance measures, a FTA identifies not only what is important but also what is unimportant. For those contributors that are unimportant and have negligible impact on the top event, resources can be relaxed with negligible impact on the top event probability. In fact, using formal allocation approaches, resources can be re-allocated to result in the same system failure probability while reducing overall resource expenditures by significant amounts. In various applications, FTA has been used to reduce resource burdens by as much as 40% without impacting the occurrence probability of the top event. Software has been developed to help carry out these resource re-allocations for large systems.

6. *Use of FTA to assist in designing a system.* When designing a system, FTA can be used to evaluate design alternatives and to establish performance-based design requirements. In using FTA to establish design requirements, performance requirements are defined and the FTA is used to determine the design alternatives that satisfy the performance requirements. Even though system specific data are not available, generic or heritage data can be used to bracket performance. This use of FTA is often overlooked, but is important enough to be discussed further in a subsequent section.

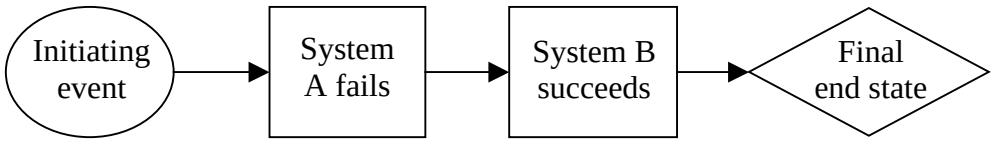
7. *Use of FTA as a diagnostic tool to identify and correct causes of the top event.* This use of FTA as a diagnostic tool is different from the proactive and preventative uses described above. FTA can be used as a diagnostic tool when the top event or an intermediate event in the fault tree has occurred. When not obvious, the likely cause

or causes of the top event can be determined more efficiently using the FTA power to prioritize contributors. The chain of events leading to the top event is identified in the fault tree, providing valuable information on what may have failed and the areas in which improved mitigation could be incorporated. When alternative corrective measures are identified, FTA can be used to objectively evaluate their impacts on the top event re-occurrence. FTA can also be an important aid to *contingency analysis* by identifying the most effective actions to be taken to reduce the impact of a fault or failure. In this case, components are set to a failed condition in the fault tree and actions are identified to minimize the impact of the failures. This contingency analysis application is often used to identify how to reconfigure a system to minimize the impact of the component failures. *Allowed downtimes* and repair times can also be determined to control the risk incurred from a component failure.

As can be seen from the above, FTA has a wide variety of uses and roles it can play in decision-making. FTA can be used throughout the life cycle of the system from design through system implementation and improvement. As the system proceeds to the end of life, its performance can be monitored to identify trends before failure occurs. When consciously used to assist decision-making, the payoffs from FTA generally far outweigh the resources expended performing the analysis.

1.6 Role of Fault Trees in a PRA

A Probabilistic Risk Assessment, or PRA, models sequences of events that need to occur in order for undesired end states to occur. A sequence of events (event sequence) is usually called an accident sequence. An example of an accident sequence is a fire that leads to catastrophic consequences because mitigation systems fail to operate. A model of a simple event sequence in a PRA is shown below.



Notice that in the above event sequence model, success of a system as well as failure of another system appears. Which particular systems fail and which succeed determine the type of end state and its associated consequences. To quantify the accident sequence, a probability for each event in the event sequence, other than the end state, needs to be determined. The probability of each event is conditional on the previous events in the sequence (e.g., the probability of system A failing is the probability of A failing given the initiating event occurs, the probability of system B succeeding is the probability of B succeeding given A fails and the initiating event occurs). If an event is independent of others in the sequence and failure data exist, the probability can be directly estimated from the data. For more complex events in the sequence, that do not have directly applicable data or that may have dependencies on other events in the sequence, such as for a system failure, a fault tree is usually constructed. The fault tree is developed to a level that encompasses the dependencies between systems or to a level where failure data exist for the basic events, whichever is lower (more detailed). The fault tree is then evaluated to determine the probability of the system failure.

Each event sequence is a logical intersection (an AND gate) of the initiating event and the subsequent events other than the end state. Available PRA software automatically carries out the operations involving this intersection using all the fault trees that are input to an event sequence. Depending upon the level of resolution, a complex PRA such as for the Space Shuttle can have tens of thousands of accident sequences involving hundreds of different fault trees. In a large analysis, the fault trees (AND gates) of each sequence are combined into a single OR gate to generate accident sequence cut sets for the entire PRA in a single analysis run. When several different end states are defined the fault trees for each individual end state are combined. Fault trees are generally the work horses of a PRA, providing causes and probabilities for all the system failures involved, as well as a framework for quantification of the sequences.

1.7 Software for Fault Tree Analysis

A number of software applications exist for FTA and new applications are continually being developed. Some applications provide the capability to draw and quantify FT models, while others provide an integrated set of PRA tools that include the capability to draw and solve FTs. It is not the purpose of this document to serve as reference for FT-related software; it is not possible to describe all FT software that is currently available and it is clearly not possible to describe software that may be available in the future. Therefore, this Handbook does not include FT software references (the one exception is the Galileo/ASSAP software [2] developed by NASA to solve the DFTs described in Chapter 8).

1.8 References

1. W. Vesely et al., *Fault Tree Handbook*, NUREG-0492, Nuclear Regulatory Commission, 1981.

2. K. Sullivan, J. Dugan and D. Coppit, “The Galileo Fault Tree Analysis Tool,” Proceedings of IEEE International Symposium of Fault Tolerant Computing, FTC-29, June 1999, pp 232-235.

2. System Logical Modeling Approaches

2.1 Success vs. Failure Approaches

The operation of a system can be considered from two standpoints: the various ways for system success can be enumerated or the various ways for system failure can be enumerated. Such an enumeration would include completely successful system operation and total system failure, as well as intermediate conditions such as minimum acceptable success. Figure 2-1 depicts the Failure/Success space concept.

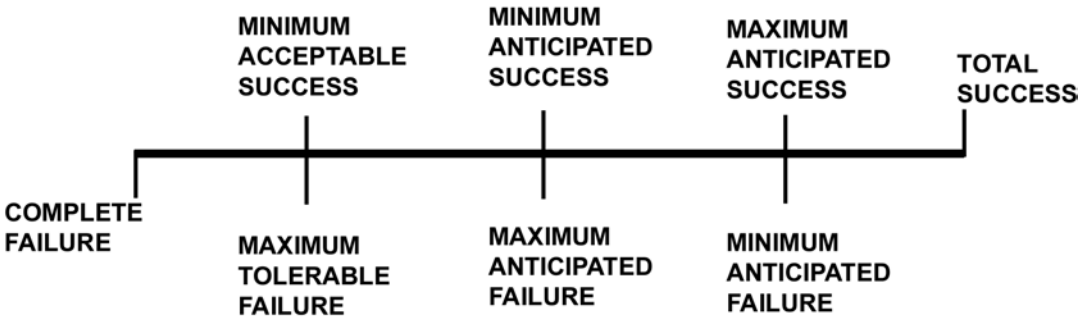


Figure 2-1. The Failure Space-Success Space Concept

It is interesting to note that certain identifiable points in success space coincide with certain analogous points in failure space. Thus, for instance, “maximum anticipated success” in success space can be thought of as coinciding with the “minimum anticipated failure” in failure space. Although the first inclination might be to select the optimistic view of our system (success) rather than the pessimistic one (failure) when considering system operation, this is not necessarily the most advantageous perspective to take.

From an analytical standpoint, there are several overriding advantages that accrue from the failure space perspective. First of all, it is generally easier to attain concurrence on what constitutes failure than it is to agree on what constitutes success. An aircraft might be desired to fly high and fast, travel far without refueling and carry a big load. When the final version of this aircraft rolls off the production line, some of these features may have been compromised in the course of making design trade-offs. Whether the vehicle is a "success" or not may very well be a matter of controversy. On the other hand, if the aircraft crashes, there will be little argument that this event constitutes system failure.

“Success” tends to be associated with the efficiency of a system, the amount of output, the degree of usefulness, and production and marketing features. These characteristics are describable by continuous variables that are not easily modeled in terms of simple discrete events, such as “valve does not open,” which characterize the failure space (partial failures, i.e., a valve opens partially, are also difficult events to model because of their continuous possibilities). Thus, the event “failure,” or in particular, “complete failure,” is generally easy to define, whereas the event “success” may be much more difficult to tie down. This fact makes the use of failure space in analysis much more valuable than the use of success space.

Another point in favor of the use of failure space is that, although theoretically the number of ways in which a system can fail and the number of ways in which a system can succeed are both infinite, from a practical standpoint there are generally more ways to success than there are to failure. Thus, purely from a practical point of view, the size of the population in failure space is less than the size of the population in success space. In analysis, therefore, it is generally more efficient to make calculations on the basis of failure space.

A final point in favor of the use of failure space is the nature of the mathematics involved in the quantification of failure models. Most failure probabilities are small (less than 0.1), which allows the use of accurate approximations when combining failure probabilities. Since success probabilities are usually close to 1.0, these approximations cannot be used, necessitating the use of complex calculations when combining success probabilities. The solution of success models is therefore much more different than the solution of failure models.

The advantageous use of the failure space when analyzing system operation has been demonstrated on numerous occasions in the past. The drawing of logic diagrams for a complex system is an expensive and time-consuming operation. When failures are considered, it may be necessary to construct only one or two system models, such as fault trees, that cover all the significant failure modes. When successes are considered, it may become necessary to construct several hundred system models covering various definitions of success. A good example of the parsimony of events characteristic of failure space was the Minuteman missile analysis. Only three fault trees were drawn corresponding to the three undesired events: inadvertent programmed launch, accidental motor ignition, and fault launch. It was found that careful analysis of just these three events provided a complete overview of the complex Minuteman system.

Consider the “mission” in Figure 2-2 referring to the transport of person X by automobile from home to the office. The desired arrival time is 8:30, but the mission will be considered marginally successful if X arrives at the office by 9:00. Arrival at 8:30 is labeled “minimum anticipated failure.” Below “minimum anticipated failure” lie a number of possible incidents that constitute minor annoyances, but which do not prevent X from arriving at the desired time. Arrival at 9:00 is labeled “maximum anticipated failure.” Between this point and “minimum anticipated failure” lie a number of occurrences that cause X’s arrival time to be delayed half an hour or less. It is perhaps reasonable to let the point “maximum tolerable failure” coincide with some accident that causes some damage to the car and considerable delay but no personal injury. Above this point lie incidents of increasing seriousness terminating in complete failure or death.

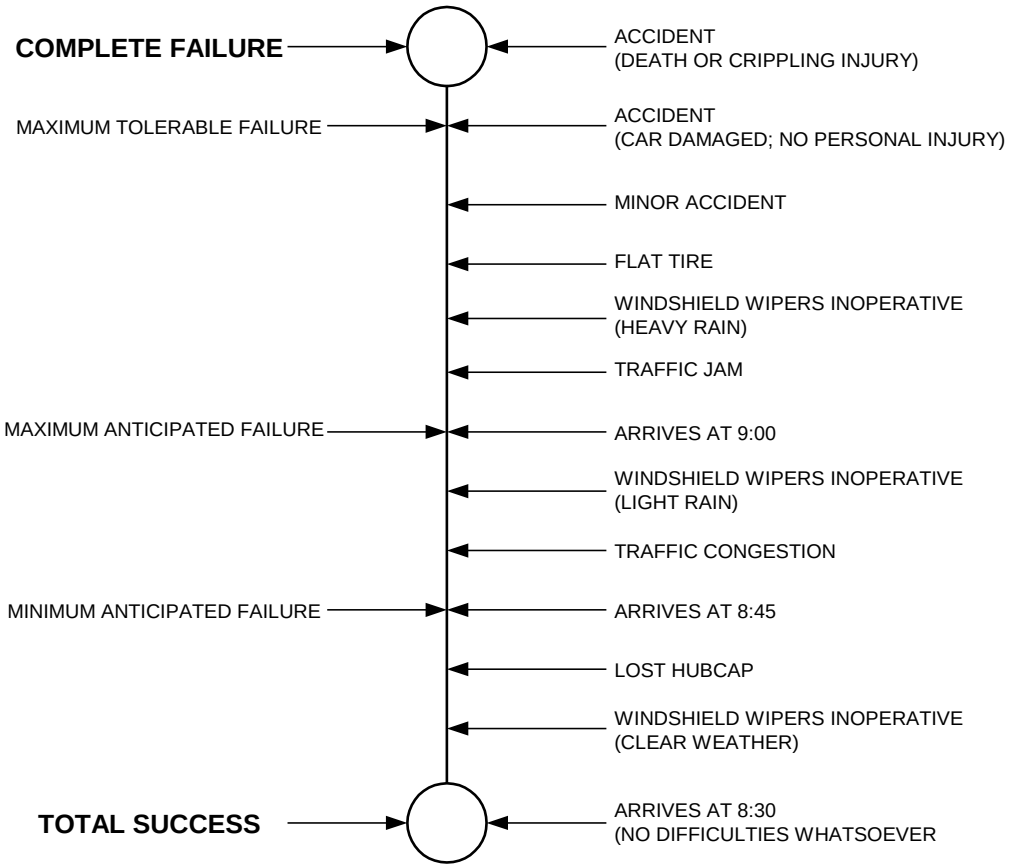


Figure 2-2. Use of Failure Space in Transport Example

Note that an event such as "windshield wipers inoperative" will be positioned along the line according to the nature of the environment at that time.

A chart such as Figure 2-2 might also be used to pinpoint events in, for example, the production of a commercial airliner. The point "minimum anticipated failure" would correspond to the attainment of all specifications and points below that would indicate that some of the specifications have been more than met. The point "maximum anticipated failure" would correspond to some trade-off point at which all specifications had not been met but the discrepancies were not serious enough to degrade the saleability of the aircraft in a material way. The point "maximum tolerable failure" corresponds to the survival point of the company building the aircraft. Above that point, only intolerable catastrophes occur. Generally speaking, but not in all cases, FTA addresses itself to the identification and assessment of just such catastrophic occurrences and complete failures.

2.2 Deductive Methods and FTA

Deduction constitutes reasoning from the general to the specific. In a *deductive* system analysis, it is postulated that the system itself has failed in a certain way, and an attempt is made to find out what modes of system or subsystem (component) behavior contribute to this failure. In common parlance this approach might be referred to as a "Sherlock Holmes" approach. Holmes, faced with given evidence, has the task of reconstructing the events leading up to the crime.

Indeed, all successful detectives and other types of investigators are experts in deductive analysis.

Typical of deductive analyses in real life are accident investigations: what chain of events caused the sinking of an "unsinkable" ship such as the Titanic on its maiden voyage? What failure processes, instrumental and/or human, contributed loss of the vertical stabilizer and to the crash of a commercial airliner into a residential area?

The principal subject of this book, Fault Tree Analysis, is an example of deductive systems analysis. In this technique, some specific system state, which is generally a failure state, is postulated, and chains of more basic faults contributing to this undesired event are built up in a systematic and logical way. The broad principles of FTA, as well as details relating to the applications and evaluation of FTs, are given in later chapters.

While deductive methods are applied to determine *how* a given system state (usually a failed state) can occur, inductive methods can be applied to determine what system states (usually failed states) are possible. Induction methods play an important role in risk and reliability analysis, particularly in the development of accident scenarios and in ensuring completeness. Inductive methods are described in the next section and in Chapter 9 and compared to FTA in Section 2.4.

2.3 Inductive Methods

Induction involves reasoning from individual cases to a general conclusion. If, in the consideration of a certain system, a particular fault or initiating condition is postulated and an attempt to ascertain the effect of that fault or condition on system operation is made, an inductive system analysis is being conducted. Thus, an inquiry might be made into how the loss of some specified control surface affects the flight of an aircraft or into how the elimination of some item in the budget affects the overall operation of a school district.

In everyday language the inductive techniques provide answers to the generic question, "What happens if…?" More formally, the process consists of assuming particular states for components, generally failed states, and then analyzing the effects on the system. More generally, a given initiating event is assumed, such as a pipe rupture, and the consequences of the event are analyzed. This more general approach is used in event trees and event sequence diagrams, which are discussed in Chapter 9.

Inductive approaches are also termed bottom-up approaches that start at the bottom, i.e., at the failure initiators and basic event initiators, and then proceed upwards to determine the resulting system effects of a given initiator. Inductive approaches thus start at a possible basic cause and then analyze the resulting effects. A set of possible causes are analyzed for their effects. Principal examples of inductive approaches are described below.

The "Parts Count" Approach

Probably the simplest and most conservative (i.e., pessimistic) assumption that can be made about a system is that any single component failure will produce complete system failure. Under

this assumption, obtaining an upper bound on the probability of system failure is especially straightforward. All the components are listed along with their estimated probabilities of failure. The individual component probabilities are then added and this sum provides an upper bound on the probability of system failure. The failure probabilities can be failure rates, unreliabilities, or unavailabilities depending on the particular application (these more specific terms will be covered later).

For a particular system, the Parts Count technique can provide a very pessimistic estimate of the system failure probability and the degree of pessimism is generally not quantifiable. The “Parts Count” technique is conservative because if critical components exist, they often appear redundantly, so that no single failure is actually catastrophic for the system. Furthermore, a component can often depart from its normal operating mode in several different ways and these failure modes will not, in general, all have an equally deleterious effect on system operation. If the relevant failure modes for the system operation are not known then it is necessary to sum the failure probabilities for all the possible failure modes. The parts count method is not discussed further but has been introduced as the simplest inductive approach where every component failure is assumed to cause system failure.

Failure Mode and Effect Analysis (FMEA)

Table 2-1 shows an FMEA constructed of a redundant system of amplifiers. In constructing the table, it is recognized that amplifiers can fail in several ways and the first task is the identification of these various failure modes. The two principal ones are "open" and "short," but suppose that the analysis has also detected 28 other modes (e.g., weak signal, intermittent ground, etc.). A short of any amplifier is one of the more critical failure modes inasmuch as it will always cause a failure of the system. The FMEA table contains the following information:

(1) Component designation
(2) Failure probability (failure rates or unavailabilities are some of the specific characteristics used)
(3) Component failure modes
(4) Percent of total failures attributable to each mode
(5) Effects on overall system, classified into various categories (the two simplest categories are “critical” and “non-critical”).

Based on prior experience with this type of amplifier, it is estimated that 90% of amplifier failures can be attributed to the “open” mode, 5% of them to the “short” mode, and the balance of 5% to the “other” modes. It is known that whenever either amplifier fails shorted, the system fails so X's are placed in the "Critical" column for these modes; “Critical” thus means that the single failure causes system failure. On the other hand, when either amplifier fails open, there is no effect on the system from the single failure because of the parallel configuration. What is the criticality of the other 28 failure modes? This example is conservative in that all the other failure modes are considered to be critical, i.e., the occurrence of any one causes system failure. The

numbers shown in the Critical column are obtained from multiplying the appropriate percentage in Column 4 by 1×10^{-3} from Column 2.*

Table 2-1. Redundant Amplifier Analysis

1	2	3	4	5 Effects	
Component	Failure Probability	Failure Mode	% Failures by Mode	Critical	Non-Critical
A	1×10^{-3}	Open	90		X
		Short	5	X (5×10^{-5})	
		Other	5	X (5×10^{-5})	
B	1×10^{-3}	Open	90		X
		Short	5	X (5×10^{-5})	
		Other	5	X (5×10^{-5})	

Based on the table, the probability of system failure from single causes can be more realistically calculated, considering now only those failure modes that are critical. Adding up the critical column, Column 5, the probability of system failure = $5 \times 10^{-5} + 5 \times 10^{-5} + 5 \times 10^{-5} + 5 \times 10^{-5} = 2 \times 10^{-4}$ is obtained. This is a less conservative result compared to 2×10^{-3} obtained from the parts count method where the critical failure modes were not separated. The difference between the two system results can be large, i.e., an order of magnitude or more, as in this example, if the critical failure modes are a small percentage of the total failure mode (e.g., 10% or less).

FMEA (and its variants) can identify, with reasonable certainty, those component failures having "non-critical" effects, but the number of possible component failure modes that can realistically be considered is limited. Conservatism dictates that unspecified failure modes and questionable effects be deemed "critical" (as in the previous example). The objectives of the analysis are to identify single failure modes and to quantify these modes; the analysis needs be no more elaborate than is necessary for these objectives.

Failure Mode Effect and Criticality Analysis (FMECA)

Failure Mode Effect and Criticality Analysis (FMECA), is essentially similar to a Failure Mode and Effects Analysis in which the criticality of the failure is analyzed in greater detail, and assurances and controls are described for limiting the likelihood of such failures. Although FMECA is not an optimal method for detecting hazards, it is frequently used in the course of a

* The notation 1×10^{-3} is scientific notation and describes 0.001, where the decimal is moved over three places to the left of 1.

system safety analysis. The four fundamental facets of such an approach are (1) Fault Identification, (2) Potential Effects of the Fault, (3) Existing or Projected Compensation and/or Control, and (4) Summary of Findings. These four facets generally appear as column headings in an FMECA layout. Column 1 identifies the possible hazardous condition. Column 2 explains why this condition is a problem. Column 3 describes what has been done to compensate for or to control the condition. Finally, Column 4 states whether the situation is under control or whether further steps should be taken.

At this point the reader should be warned of a most hazardous pitfall that is present to a greater or lesser extent in all these inductive techniques: the potential of mistaking form for substance. If the project becomes simply a matter of filling out forms instead of conducting a proper analysis, the exercise will be completely futile. For this reason it might be better for the analysis not to be restricted to any prepared formalism. Another point: if the system is at all complex, it is foolhardy to imagine that a single analyst can conduct a correct and comprehensive survey of all system faults and their effects on the system. These techniques call for a well-coordinated team approach. Moreover, FMEAS and FMECAs analyze single component faults and their system effects and do not consider combinations of component faults.

Preliminary Hazard Analysis (PHA)

The techniques described so far have been, for the most part, system oriented, i.e., the effects are faults on the system operation. The subject of this section, Preliminary Hazard Analysis (PHA), is a method for assessing the potential hazards posed, to plant personnel and other humans, by the system.

The objectives of a PHA are to identify potential hazardous conditions inherent within the system and to determine the significance or criticality of potential accidents that might arise. A PHA study should be conducted as early in the product development stage as possible. This will permit the early development of design and procedural safety requirements for controlling these hazardous conditions, thus eliminating costly design changes later on.

The first step in a PHA is to identify potentially hazardous elements or components within the system. This process is facilitated by engineering experience, the exercise of engineering judgment, and the use of numerous checklists that have been developed from time to time. The second step in a PHA is the identification of those events that could possibly transform specific hazardous conditions into potential accidents. Then the seriousness of these potential accidents is assessed to determine whether preventive measures should be taken.

Various columnar formats have been developed to facilitate the PHA process. Perhaps the simplest is:

Column (1) Component/subsystem and hazard modes
Column (2) Possible effects
Column (3) Compensation and control
Column (4) Findings and remarks

Fault Hazard Analysis (FHA)

Another method, Fault Hazard Analysis (FHA), was developed as a special purpose tool for use on projects involving many organizations, one of which is supposed to act as integrator. This technique is especially valuable for detecting faults that cross organizational interfaces. Even though FHA is generally not used now per se, FHA concepts and approaches are used in certain extended FMEAs and FMECAs. The FHA approach considers the following basic causes and effects, which can be arranged in columns and which characterize this form of inductive approach.

Column (1) Component identification
Column (2) Failure probability
Column (3) Failure modes (identify all possible modes)
Column (4) Percent failures by mode
Column (5) Effect of failure (traced up to some relevant interface)
Column (6) Identification of upstream component that could command or initiate the fault in question
Column (7) Factors that could cause secondary failures (including threshold levels). This column should contain a listing of those operational or environmental variables to which the component is sensitive.
Column (8) Remarks

What is different for FHA is the consideration of the extra information given in Columns 6 and 7. Column 6 identifies possible command or interface failures. Column 7 identifies secondary failures that are failures outside the design envelope. As will become apparent in later chapters, Columns 6 and 7 have special significance for the fault tree analyst.

Double Failure Matrix (DFM)

The previous techniques concerned themselves with the effects of single failures. An inductive technique that also considers the effects of double failures is the Double Failure Matrix (DFM); its use is feasible systems with small numbers of redundant components. The DFM approach is useful to discuss since it provides an extension of inductive approaches from single failure causes to multiple failure causes. This is a significant enhancement to FMEA and FMECA approaches. To more effectively apply the DFM approach faults, including multiple faults, are first categorized according to the severity of the system effect. A basic categorization that originated in MIL STD 882 and that is still used is shown in Table 2-2.

Table 2-2. Fault Severity Categories and Corresponding System Effects

Fault Category	Effect on System
I	Negligible
II	Marginal
III	Critical
IV	Catastrophic

More complete definitions of the system effects are:

(I) Negligible–loss of function that has no effect on system.
(II) Marginal-this fault will degrade the system to some extent but will not cause the system to be unavailable; for example, the loss of one of two redundant pumps, either of which can perform a required function.
(III) Critical-this fault will completely degrade system performance; for example, the loss of a component which renders a safety system unavailable.
(IV) Catastrophic-this fault will produce severe consequences which can involve injuries or fatalities; for example, catastrophic pressure vessel failure.

The categorization will depend on the conditions assumed to exist previously, and the categorizations can change as the assumed conditions change. For example, if one pump is assumed failed, then the failure of a second redundant pump is a critical failure.

The above fault categorizations can be refined in many ways. For example, six fault categories are shown in Table 2-3. These fault categories were originally used in the NERVA project but have been used in more recent times with slight modifications

Table 2-3. Extended Fault Severity Categories

Fault Category	Effect on system
I	Negligible
IIA	A second fault event causes a transition into Category III (Critical)
IIB	A second fault event causes a transition into Category IV (Catastrophic)
IIC	A system safety problem whose effect depends upon the situation (e.g., the failure of all backup onsite power sources, which is no problem as long as primary, offsite power service remains on)
III	A critical failure and mission must be aborted
IV	A catastrophic failure

To illustrate the application of DFM, consider the simple subsystem shown in Figure 2-3. In this figure, the block valves can operate only as either fully open or fully closed, whereas the control valves are proportional valves which may be partially open or partially closed.

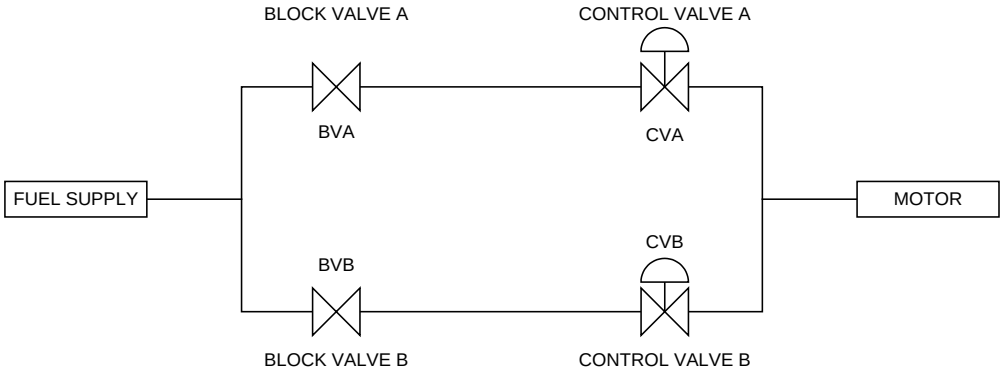


Figure 2-3. Fuel System Schematic

Let two fault states for this system be defined and categorized as follows:

Fault State	Category
No flow when needed	IV
Flow cannot be shut off	III

All possible component failures and their fault categories are now considered. For instance, Block Valve A (BVA) failed open is identified as Category IIA because, if Control Valve A (CVA) is also failed open, Category III results. BVA failed closed is identified as Category IIB because, if either BVB or CVB is also failed closed, Category IV results. This type of analysis is conveniently systematized in the Double Failure Matrix shown in Table 2-4.

For illustrative purposes the entire matrix has been filled in; for a first-order analysis the concern would be only with the main diagonal terms, to wit, the single failure states. Note that if BVA is failed open, there is only one way in which a second failure can cascade into Category III; namely, CVA must be failed open too. In contrast, if BVA is failed closed, Category IV can result if either BVB or CVB is also failed closed, which is why "Two Ways" is given in Table 2-4. Similar considerations apply to the single failures of CVA, BVB and CVB and this important additional information has been displayed in the principal diagonal cells of the matrix.

Now concentrating only on single failures, a hazard category count is conducted as the following table shows:

Hazard Category	Number of Ways of Occurring
IIA	4
IIB	8

Table 2-4. Fuel System Double Failure Matrix

		BVA		CVA		BVB		CVB	
		Open	Closed	Open	Closed	Open	Closed	Open	Closed
BVA	Open	(One Way) IIA		III	IIB	IIA	IIA or IIB	IIA	IIA
	Closed		(Two Ways) IIB	IIB	IIB	IIA or IIB	IV	IIA or IIB	IV
CVA	Open	III	IIB	(One Way) IIA		IIA	IIA or IIB	IIA	IIA or IIB
	Closed	IIB	IIB		(Two Ways) IIB	IIA	IV	IIA or IIB	IV
BVB	Open	IIA	IIA or IIB	IIA	IIA or IIB	(One Way) IIA		III	IIB
	Closed	IIA or IIB	IV	IIA or IIB	IV		(Two Ways) IIB	IIB	IIB
CVB	Open	IIA	IIA or IIB	IIA	IIA or IIB	III	IIB	(One Way) IIA	
	Closed	IIA or IIB	IV	IIA or IIB	IV	IIB	IIB		(Two Ways) IIB

How can this information be used? One application would be a description and subsequent review of how these hazard categories are controlled or are insured against. Another application would be a comparison between the configuration of valves shown in Figure 2-2 and an alternative design. The DFM is a useful inductive approach for redundancies but is limited in its capability in being able to handle small combinations, especially double failures, and only a limited number of combinations can be assessed. Software allows more combinations and multiple failures beyond two to be assessed. However constraints still must be placed on the combinations considered.

Reliability Block Diagram (RBD)

A reliability block diagram is an inductive model wherein a system is divided into blocks that represent distinct elements such as components or subsystems. These elemental blocks are then combined according to system-success pathways. RBDs are generally used to represent active elements in a system, in a manner that allows an exhaustive search for and identification of all pathways for success. Dependencies among elements can be explicitly addressed.

Initially developed top-level RBDs can be successively decomposed until the desired level of detail is obtained. Alternately, series components representing system trains in detailed RBDs can be logically combined, either directly or through the use of FTs, into a supercomponent that is then linked to other supercomponents to form a summary model of a system. Such a representation can sometimes result in a more transparent analysis.

An example RBD for the fuel system depicted in Figure 2-3 is shown in Figure 2-4. Separate blocks representing each system element (fuel supply, block valves, control valves and motor) are structurally combined to represent both potential flow paths through the system. The model is solved by enumerating the different success paths through the system and then using the rules of Boolean algebra to continue the blocks into an overall representation of system success.

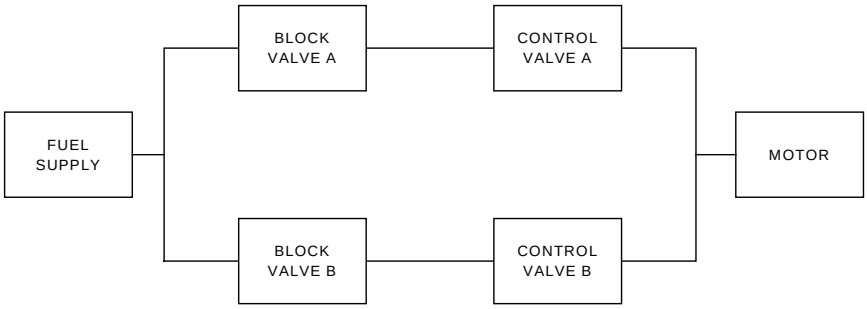


Figure 2-4. Reliability Block Diagram of a Fuel System.

Software exists to convert RBDs into FTAs and vice versa. While these conversions result in logically equivalent models, the logical representation in the conversion may not be as clear as in the original model.

2.4 Comparison of FTA with Inductive Methods

The basic difference between FTA and inductive methods is the direction of the analysis. A FTA starts with the undesired event and traces backward to the necessary and sufficient causes. The fault tree ends with the initiating basic events and failures that are identified as the primary causes. An inductive approach starts with an initiating cause and traces forward the resulting consequences. This forward stepping is repeated for different selected initiating causes. The end consequences can vary depending on the initiating cause. FTA is thus the appropriate analysis to carry out if a given undesired event is defined and the goal is to determine its basic causes.

Inductive analysis is the appropriate analysis to carry out if a given set of initiating causes are identified and the goal is to determine the resulting consequences. A PRA uses both inductive and deductive approaches. Initiating events are typically identified using a deductive approach called a Master Logic Diagram (MLD). This concept is described in Reference 1. For initiating events that can progress to multiple failure states that must be separately tracked (i.e., Space Shuttle LOV), PRA uses an inductive tool such as event trees not only to determine the resulting consequences, but also to enumerate all possible accident scenarios. A system analysis generally evaluates the causes of a defined undesirable system event, such as system failure. Hence, FTA is the appropriate analysis tool for a system analysis.*

* It is also important to note that, in general, performing an FTA alone should not be mistakenly perceived as an alternative to or substitute for a complete PRA. In particular, a complete and scenario-based PRA requires more than just a straight forward FTA.

In general, both deductive and inductive approaches must be employed to get a complete set of accident sequences. The deductive approach has the benefit of focusing the analysis on the undesired event while the inductive approach is useful in assuring that the analysis is broad enough to encompass all possible scenarios.

There are of course overlaps between FTA and inductive analysis. As discussed, fault trees are used in a PRA to analyze specific system failures in the event sequences. Multiple fault trees can be, and often are, constructed for different undesired events to evaluate a spectrum of consequences. In these applications, the basic difference between FTA and inductive analysis remains in the direction of the analysis, the FTA as a backward analysis and the inductive analysis as a forward analysis.

2.5 References

1. *Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners*, NASA, Version 1.1, August 2002.

3. Fault Tree Analysis

3.1 Steps in Carrying Out a Fault Tree Analysis

A successful FTA requires the following steps be carried out:

1. Identify the *objective* for the FTA.
2. Define the *top event* of the FT.
3. Define the *scope* of the FTA.
4. Define the *resolution* of the FTA.
5. Define *ground rules* for the FTA.
6. *Construct* the FT.
7. *Evaluate* the FT.
8. *Interpret* and present the results.

The first five steps involve the problem formulation for an FTA. The remaining steps involve the actual construction of the FT, the evaluation of the FT, and the interpretation of the FT results. While most of the steps are performed sequentially, steps 3-5 can proceed concurrently. It is not uncommon for steps 4 and 5 to be modified during steps 6 and 7. The interrelationship of the eight steps are shown in Figure 3-1. The feedback is indicated in the figure.

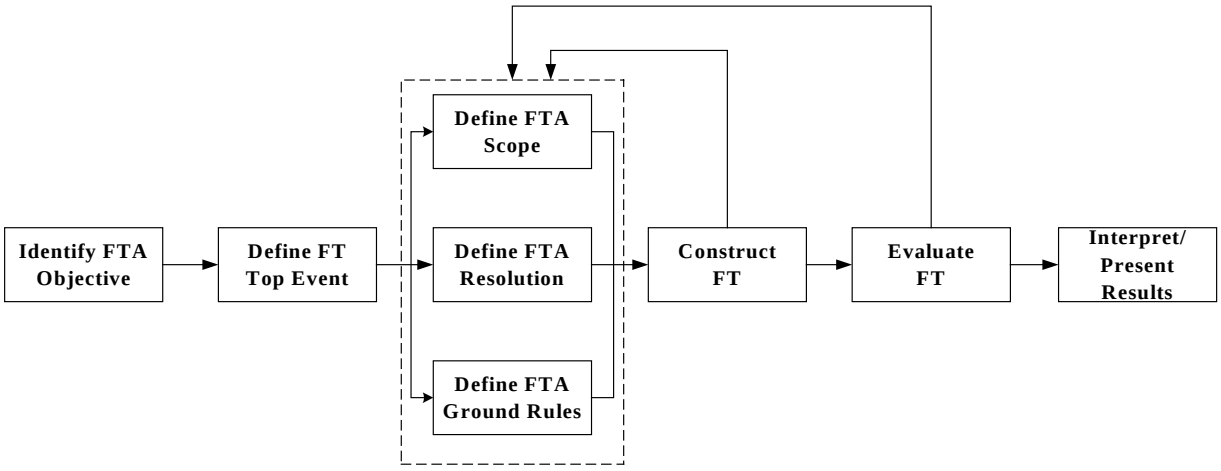


Figure 3-1. Fault Tree Analysis Steps.

The first step for a successful FTA is to define the objective of the FTA. This may seem obvious; however, there have been many cases where a FTA is performed, but the analysis does not satisfy the objective of the decision maker or manager who commissioned it. To be successful the objective should be phrased in terms of a failure of the system to be analyzed. For example if the general objective is to evaluate different designs for a mission then the particular failure that characterizes mission failure and that will be analyzed to evaluate the designs needs to be identified.

Once the objective is defined in this way then the top event of the FT is also defined (Step 2). The top event of the FT is the event for which the failure causes will be resolved and the failure probability determined. The top event defines the failure mode of the system that will be

analyzed. An example would be the benign shutdown of an engine. Sometimes the objective may entail defining and analyzing more than one failure. In this case separate top events are then defined. In particular, for a given mission there may be several objectives and the resulting fault tree might be very different depending on the particular objective chosen for the analysis. For example, for a manned launcher the objective might be loss of vehicle (LOV), loss of crew (LOC) or loss of mission (LOM). For an unmanned Mars sample return mission the objective might be sample containment assurance (CA) or mission success (MS). In each case the resulting fault trees would likely be different even for the same mission. If the mission has different phases, then a separate top event for each phase may need to be defined. Phased mission analysis will be discussed in a subsequent section.

In Step 3, the scope of the analysis is defined. The scope of the FTA indicates which of the failures and contributors will be included and which will not be included. The scope of the FTA also includes the particular design version and historical time period relevant to the system that will be analyzed. Finally, the scope includes the boundary conditions for the analysis. The boundary conditions include the initial states of the components and the assumed inputs to the system. The FT represents a snapshot of the system at a given time for a given configuration and boundary. For example, assume the failure of a flight control system is being analyzed. In defining the scope, the version of the system to be analyzed is identified, the modes of operation defined, the component failures to be considered are indicated, and the interfaces to system (e.g., support systems, actuation signals) that will be modeled for their failures or that will be assumed to not fail are identified.

In Step 4 of the process, the resolution of the FTA is defined. The resolution is the level of detail to which the failure causes for the top event will be developed. If the top event is a functional failure of the system, such as failure to operate or inadvertent shutdown, then the top event is generally resolved to the major components in the systems. Examples of major components are valves, pumps, and control modules. If the top event is a phenomenological failure such as a catastrophic explosion of an engine then the resolution is the level of detail to which the causes of the explosion will be modeled. The development of a quantitative model is based on the need to get the best possible estimate for the top event probability, considering the data and other information that are available. Fault trees are developed to a level of detail where the best failure probability data are available. Further resolution of the system is necessary when decisions about subcomponents or support systems are being made, or when an event cannot be shown to be independent of others in the analysis (e.g., a system that has actuation signals or power in common with other systems). The FT can, and often is, developed to a level of detail that is below the level where data is available to estimate the basic event probabilities or to where the risk discrimination no longer matters. This is often unnecessary. The FTs that will be illustrated in later sections demonstrate resolution levels for different types of systems and top events.

In Step 5 any ground rules for the FTA are defined. These ground rules include the procedure and nomenclature by which events and gates are named in the FT. The naming scheme used is very important in creating an understandable FT. Examples of naming schemes are given for the FTs that will be illustrated. Ground rules can also be given for the manner in which specific failures are modeled in the FT. These modeling ground rules are useful in providing consistency among different FTs especially when different individuals are developing them. The modeling ground rules can include the manner in which specific component failures, human errors and

common cause failures are to be modeled. In later sections some of the modeling ground rules that can be used will be described. The FTs that will be presented will also illustrate some modeling ground rules that have been applied.

Step 6 involves the actual construction of the FT. The subsequent sections describe in detail the thinking and logic that is involved in constructing the fault tree from the system schematics and descriptions. The symbols that are used in the fault tree to represent the relationships between events are also described.

Step 7 involves the evaluation of the FT. The evaluation includes both a qualitative and quantitative evaluation. The qualitative evaluation provides information on the minimal cut sets for the top event. Success sets may also be identified that guarantee prevention of the top event. The nature of the basic events and the number of basic events in the combined sets give important information about the top event occurrence. Cut sets are usually sorted by cut set order (the number of events in a cut set) to provide information on the combinations of basic events that can result in the top event. The quantitative evaluation produces not only the probability of the top event but also the dominant cut sets that contribute to the top event probability, as well as quantitative importance of each basic event contributing to the top event. Cut sets in this case are sorted by probability, and low probability cut sets are truncated from the analysis. (Sometimes the top event probability is calculated without solving for cut sets when calculational efficiency is required.) Different quantitative importances are determined for different applications. Sensitivity studies and uncertainty evaluations provide further key information.

Finally, Step 8 involves the interpretation and presentation of the results. Emphasis is placed upon the interpretation and not simply on the presentation. Many a FTA has failed to have significant impact because the results were simply documented in a report. The results must be interpreted to provide tangible implications, especially concerning the potential impact upon the objective. If a decision maker or manager is given only a set of numerical values and a handful of jargon then the analysis may have little impact and will likely dissuade managers from attempting a future FTA!

3.2 Basic Paradigm in Constructing a Fault Tree

The basic paradigm in constructing a fault tree is “think small”, or more accurately “think myopically.” For each event that is analyzed, the *necessary and sufficient immediate events* (i.e., the most closely related events) that result in the event are identified. The key phrase is “the necessary and sufficient immediate events.” The analysis doesn’t jump to the basic causes of the event. Instead, a small step is taken and the immediate events that result in the event are identified. This taking of small steps backwards assures that all of the relationships and primary causes will be uncovered. This smallest stepping also provides the analyst with the insight into the relationships that are necessary and sufficient for the occurrence of the top event of the fault tree. This backward stepping ends with the basic causes identified that constitute the resolution of the analysis.

Subsequent sections of this handbook will describe the backward, immediate stepping involved in constructing a fault tree. However, a simple example will illustrate the paradigm. Consider

the situation where no water flows from a faucet. The immediate causes of this are not that the water company has turned off the water or that water line outside the home has ruptured (although these may eventually be the primary causes), the immediate causes are either that the faucet is failed (i.e., plugged) or that no water is being supplied to the faucet. If the faucet is checked and found to be operable then the water line to the faucet is checked to see if it is plugged or ruptured. If it is not then the next obstacle in the line, such as an upstream valve, is checked. If no water is flowing from the valve then the immediate causes are that the valve is failed or no water is being supplied to the valve. This backward immediate stepping is carried out until the primary cause or causes are identified. This thought process illustrates the paradigm of a fault tree analysis.

3.3 Boundaries of the Analysis

As with any modeling technique the boundaries of a FTA must be defined. This should be initially completed before the fault tree is constructed and must be documented. As the fault tree is being constructed, the boundaries may change and these changes must be documented. If a system failure is analyzed as the undesired event then defining the boundary of the analysis involves defining the boundary of the system that will be analyzed. Interfaces to the system such as power sources or water supplies are typically included in an analysis and are therefore within the analysis boundary. If they are excluded from the analysis then their states need to be defined in order to define the inputs to the components that are analyzed.

More generally, defining the boundaries of the analysis involves defining what is in the analysis and what is out of the analysis. What is in the analysis will be those contributors and events whose relationship to the top undesired event will be analyzed. What is out of the analysis will be those contributors that are not analyzed. There will be contributors that are at the boundary in the sense that they affect the contributors that are analyzed. The states of these interfaces need to be defined in terms of what is assumed in terms of inputs to the contributors being analyzed.

It is good practice to always show the interfaces and their assumed states on the fault tree. This serves to document the boundary of the analysis. As important, is it allows the interfaces to be developed at a later stage or allows models of the interfaces that are developed elsewhere to be joined to the fault tree.

3.4 Definition of the Top Event

The top event of the fault tree directs all of the rest of the analysis. If the top event is incorrectly defined (and this happens a surprising number of times) then the FTA will be incorrect, which can result in wrong decisions being made. There have been a surprising number of cases where the top event of the fault tree was defined incorrectly because the analyst thought it was the correct definition but did not check with the decision maker or correlate the definition with the objectives of the program. This is why it is extremely important to define and understand the objectives of the analysis and the problem to be solved. It is often fruitful to define several potential top events and then decide the appropriate one or ones based on consultation with the decision maker and others involved.

In defining the top event, it is important to define the event in terms of the specific criteria that define the occurrence of the event. Generally to do this for a system failure, the system success criteria are first defined. Then failure of the system is defined as the failure to satisfy the given success criteria. For example if the success criteria for a given cooling system is that two of three pumps start and operate for 12 hours then failure of the system is that two pumps fail to start or fail to continue to operate for 12 hours. This serves as the top event definition. The initial state of the system needs also to be defined as part of the top event or as part of the scope of the analysis. These points can be summarized as follows:

1. To define the top event, define the criteria for the occurrence of the event. For a system failure, first define the system success criteria.
2. Assure that the top event is consistent with the problem to be solved and the objectives of the analysis.
3. If unsure of the top event, define alternative definitions that cover the top event and assess the applicability of each one.

3.5 Faults vs. Failures

A distinction is made here between the rather specific word “failure” and the more general word “fault.” As an example of the distinction, consider a relay. If the relay closes properly when a voltage is applied across its terminals, this is a relay “success.” If, however, the relay fails to close under these circumstances, this is a relay “failure.” Another possibility is that the relay closes at the wrong time due to the improper functioning of some upstream component. This is clearly not a relay failure; however, untimely relay operation may well cause the entire circuit to enter into an unsatisfactory state. An occurrence like this is referred to here as a “fault” so that, generally speaking, all failures are faults but not all faults are failures. Failures are basic abnormal occurrences, whereas faults are “higher order” or more general events.

Consider next a bridge that is supposed to open occasionally to allow the passage of marine traffic. Suddenly, without warning, one leaf of the bridge flips up a few feet because of a command from the operator. This is not a bridge failure because it is supposed to open on command and it does. However, the event is a fault because the bridge mechanism responded to an untimely command issued by the bridge attendant. Thus, the attendant is part of this “system,” and it was his untimely action that caused the bridge fault.

The proper definition of a fault requires the specification of not only what the undesirable component state is but also when it occurs. These “what” and “when” specifications should be part of the event descriptions which are entered into the fault tree.

A fault may be repairable or not, depending on the nature of the system. Under conditions of no repair, a fault that occurs will continue to exist. In a repairable system a distinction must be made between the occurrence of a fault and its continued existence. This distinction is of importance only in fault tree quantification (to be discussed in a later chapter). From the standpoint of constructing a fault tree only the phenomenon of occurrence is of concern.

3.6 Failure Mechanism, Failure Mode, and Failure Effect

The definitions of system, subsystem, and component are relative, and depend upon the context of the analysis. A "system" is the overall structure being considered, which in turn consists of subordinate structures called "subsystems," which in turn are made up of basic building blocks called "components."

For example, in a fault tree of the Space Shuttle Solid Rocket Booster (SRB) the Thrust Vector Control (TVC) may be referred to as a system. A subsystem is then, for example, the Auxiliary Power Unit (APU) of the TVC. A component is then the Fuel Pump of the APU. In a particular analysis, definitions of system, subsystem, and component are generally made for convenience in order to provide a hierarchy for and boundaries to the problem. These definitions are also used in the naming scheme developed for the fault tree. A key aspect of FTA is that the elements of a system are viewed in terms of their function—artificial systems boundaries are largely ignored.

In constructing a fault tree, the basic concepts of *failure effects*, *failure modes*, and *failure mechanisms* are important in determining the proper interrelationships among the events. When failure effects are addressed, the concern is why a particular failure is of interest, i.e., what are its effects (if any) on the system? When the failure modes are detailed, exactly what aspects of component failure are of concern? When failure mechanisms are listed, how can a particular failure mode occur? Failure mechanisms are thus the means by which failure modes occur, which in turn are the effects of more basic causes. Alternatively, failure mechanisms produce failure modes, which, in turn, have certain effects on system operation.

To illustrate these concepts consider a system that controls the flow of fuel to an engine. See Table 3-1. The subsystem of interest consists of a valve and a valve actuator. Various events that can occur can be classified from either the system, subsystem, or component perspective. Some of the events are given in the left-hand column of the table below. For example, "valve unable to open" is a mechanism of subsystem failure, a mode of valve failure, and an effect of actuator failure.

Table 3-1. Fuel Flow System Failure Analysis

Description of Event	System	Subsystem	Valve	Actuator
No flow from subsystem when required	Mechanism	Mode	Effect	
Valve unable to open		Mechanism	Mode	Effect
Binding of actuator stem			Mechanism	Mode
Corrosion of actuator stem				Mechanism

To make the mechanism-mode-effect distinction clearer, consider the simple system of a doorbell and its associated circuitry from the perspective of the system, the subsystem, and the component designer. The system is shown schematically in Figure 3-2.

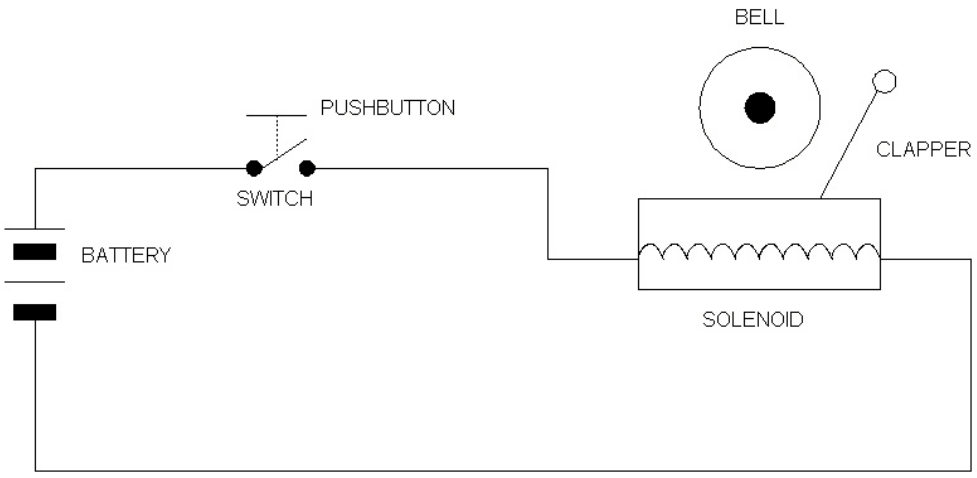


Figure 3-2. Doorbell and Associated Circuitry

From the viewpoint of the system designer, the system failure modes are:

(1) Doorbell fails to ring when button is pushed.
(2) Doorbell inadvertently rings when button is not pushed.
(3) Doorbell fails to stop ringing when push button is released.

If the system designer now sat down and made a list of the failure mechanisms causing the failure modes of interest, a list would be generated that corresponded to the failure modes of the subsystem designer who actually procures the switch, bell-solenoid unit, battery, and wires. These are:

(1) Switch - (a) fails to make contact (including an inadvertent open)
(b) fails to break contact
(c) inadvertently closes
(2) Bell-solenoid unit-fails to ring when power is applied (includes failure to continue ringing with power applied)
(3) Battery-low voltage condition
(4) Wire-open circuit or short circuit.

It is emphasized again that this last list constitutes failure mechanisms for the system designer and failure modes for the subsystem designer. It is also a list of failure effects from the standpoint of the component designer. Try to imagine what sort of list the component designer would make. See Table 3-2.

Table 3-2. Doorbell Failure Analysis

Failure Effect	Failure Mode	Mechanism
Switch fails to make contact	▪ Contacts broken ▪ High contact resistance	▪ Mechanical shock ▪ Corrosion
Bell-solenoid unit fails to ring	▪ Clapper broken or not attached ▪ Clapper stuck ▪ Solenoid link broken or stuck ▪ Insufficient magneto-motive force	▪ Shock ▪ Corrosion ▪ Open circuit in solenoid ▪ Short circuit in solenoid
Low voltage from battery	▪ No electrolyte ▪ Positive pole broken	▪ Leak in casing ▪ Shock

The system failure modes constitute the various types of system failure. In fault tree terminology these are the “top events” that the system analyst can consider. The analyst will select one of these top events and investigate the immediate causes for its occurrence. These immediate causes will be the immediate failure mechanisms for the particular system failure chosen, and will constitute failures of certain subsystems. These latter failures will be failure modes for the subsystem designer and will make up the second level of the fault tree. Proceeding, step by step, in this “immediate cause” manner the component failures are arrived at. These components are the basic causes or so-called “basic events” defined by the limit of resolution of the tree. Care should be taken to identify cases where faults in other systems can cause failures of a component, and that those faults are investigated.

From the component designer's point of view, all of the subsystem and system failures higher in the tree represent failure effects; that is, they represent the results of particular component failures. The component designer's failure modes would be the component failures themselves. If the component designer were to construct a fault tree, any one of these component failures could constitute a suitable top event. In other words, the component designer's "system" is the component itself. The lower levels of the designer's fault tree would consist of the mechanisms or causes for the component failure. These would include quality control effects, environmental effects, etc., and in many cases would be beneath the limit of resolution of the system designer’s fault tree, but need to be investigated none-the-less since they may be dominant source of overall system failure. FTA is powerful because it looks beyond system boundaries and into the role of a subsystem or component in the overall design.

3.7 Success Path Models

For the most part failures have been discussed and failures will remain the primary focus of this document. However, instead of working in “failure space,” work can be performed equivalently in “success space.” A brief example of the equivalence will be provided before the failure space approach discussion is continued.

Consider the configuration of two valves in parallel shown in Figure 3-3. This system may be analyzed either by a consideration of single failures (assuming the probabilities of multiple failures are deemed negligible) or by a consideration of "success paths." Consider the former case first.

The system requirements are as follows:

(1) The operation involves two phases;
(2) At least one valve must open for each phase;
(3) Both valves must be closed at the end of each phase.

The two relevant component failure modes are valve fails to open on demand and valve fails to close on demand. For purposes of the analysis, assume the following probability values for each of these failure modes:

P(valve does not open) = 1×10^{-4} for either phase
P(valve does not close) = 2×10^{-4} for either phase

where the symbol "P" denotes probability, and the valves are assumed to be identical.

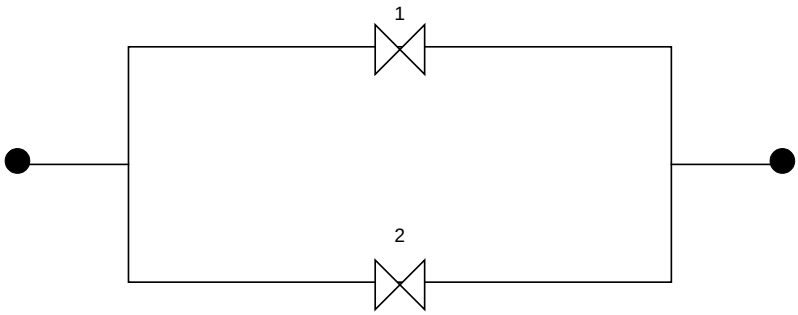


Figure 3-3. Redundant Configuration of Two Valves

The single failure analysis of the system can be tabulated as in Table 3-3.

Table 3-3. Single Failure Analysis for Redundant Valve Configuration.

COMPONENT	FAILURE MODE	FAILURE EFFECT	PROBABILITY OF OCCURRENCE (F)
Valve #1	Failure to open Failure to close	- System Failure	4×10^{-4} (either phase)
Valve #2	Failure to open Failure to close	- System Failure	4×10^{-4} (either phase)

The system failure probability is 8×10^{-4}.

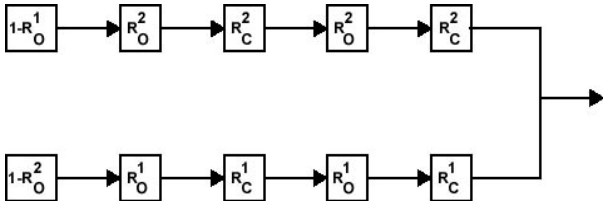
Now consider whether this result can be duplicated by considering the possible successes. There are three success paths which can be identified both notationally and schematically. If

$\mathbf{R}_O^i$ denotes "valve i opens successfully," and $\mathbf{R}_C^i$ denotes "valve i closes successfully," and P(Path i) denotes the success probability associated with the ith success path, the following diagram can be developed:

Path 1: Both valves function properly for both cycles.

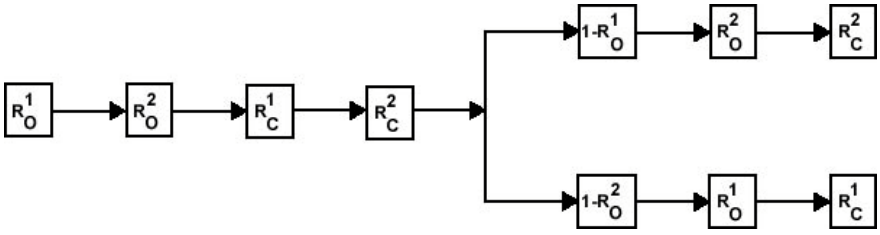
$$P(\text{Path 1}) = (R_O R_C)^4$$

Path 2: One valve fails open on the first cycle but the other valve functions properly for both cycles.



$$P(\text{Path 2}) = 2(1-R_O)(R_O R_C)^2$$

Path 3: One valve fails to open on the second cycle but the other valve functions properly for both cycles.



$$P(\text{Path 3}) = 2(1-R_O)(R_O R_C)^3$$

Numerically, system reliability is given by

$$\begin{aligned} R_{SYSTEM} &= (R_O R_C)^4 + 2(1-R_O)(R_O R_C)^2 + 2(1-R_O)(R_O R_C)^3 \\ &= 0.99880027 + 0.00019988 + 0.00019982 \\ &= 0.99919997 \cong 1 - 8\times10^{-4} \end{aligned}$$

which is essentially the same result as before. However, it should be observed that the failure approach is considerably less laborious.

3.8 Cut Sets and Path Sets

When a fault tree is evaluated then cut sets, which can also be termed failure sets, are obtained. A cut set is a set of basic events, which if they all occur, will result in the top event of the fault tree occurring. Since the basic events are the bottom, primary events of the fault tree, a cut set relates the basic events directly to the top event. A minimal cut set, informally termed a minimal

failure set, is a smallest set of basic events, which if they all occur will result in the top event occurring. The set is minimal in that if any of the events do not occur then the top event will not occur by this combination of basic events. A given fault tree will have a finite number of unique minimal cut sets. The minimal cut sets identify all the distinct ways the top event can occur in terms of the basic events. A method for obtaining minimal cut sets and their uses will be discussed in later chapters.

The logical complement of the cut sets are the path sets of a success tree. A path set, which can also be termed a success set, is a set of events, which if they all do not occur, will result in the top event not occurring. A path set thus relates the nonoccurrence of basic events to the top event nonoccurrence. A minimal path set, informally termed a minimal success set, is a smallest number of events, which if they do not occur, will result in the nonoccurrence of the top event. The set is minimal in that if any of the events occur then the nonoccurrence of the top event cannot be guaranteed by this set of events. The minimal path sets of a success tree identify all the unique ways the top event can be assured to not occur.

4. The Fault Tree Model

4.1 Symbology—The Building Blocks of the Fault Tree

A typical fault tree as shown in Figure 4-1 is composed of a number of symbols which are described in detail in the remaining sections of this chapter and are summarized for the reader's convenience in Table 4-1.

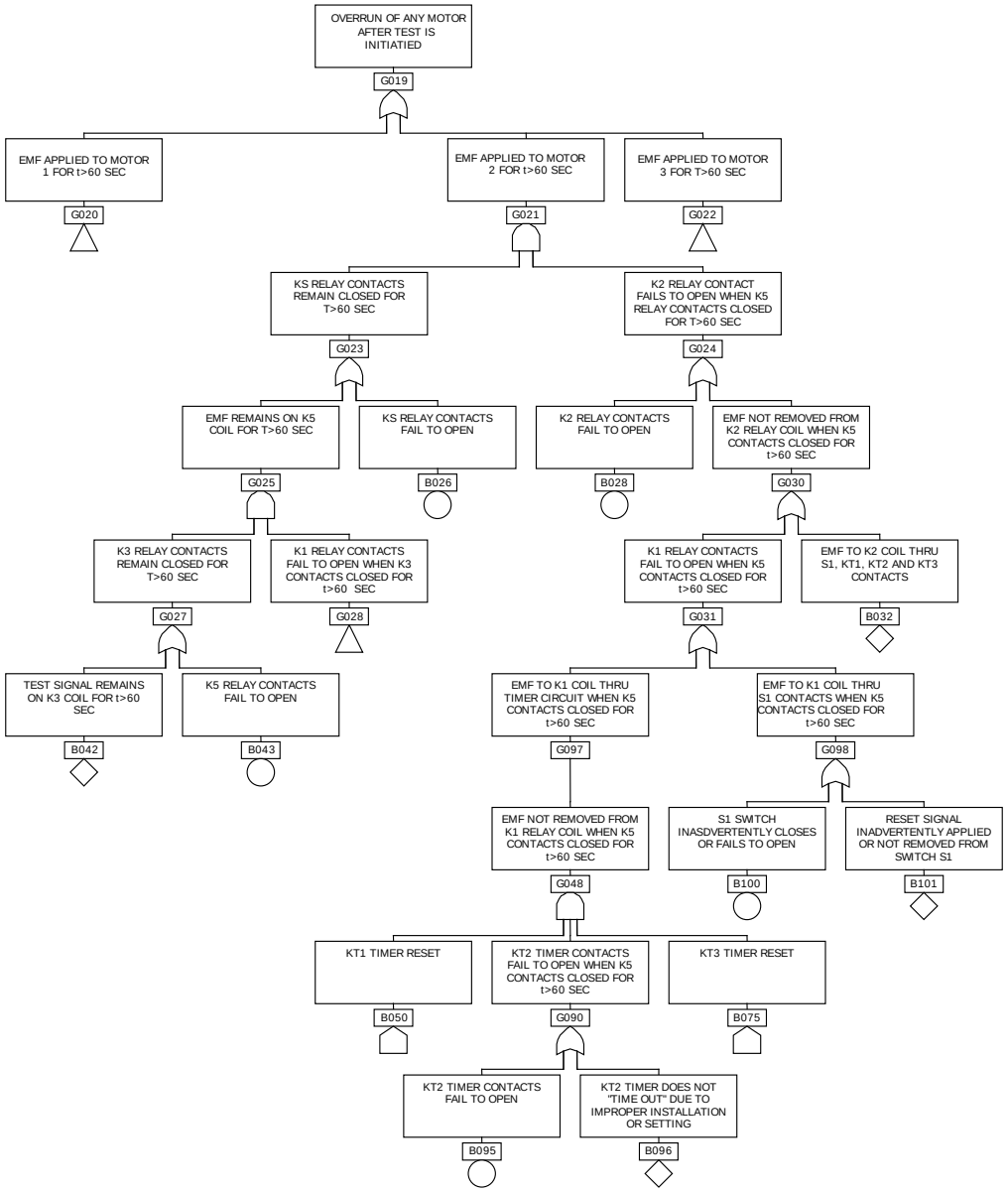
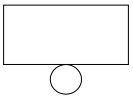
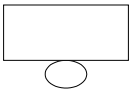
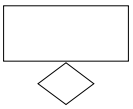
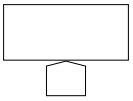
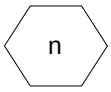
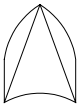
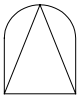
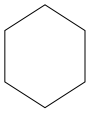
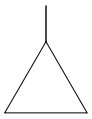


Figure 4-1. Typical Fault Tree

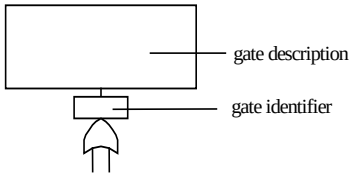
Table 4-1. Fault Tree Symbols

PRIMARY EVENT SYMBOLS	
	BASIC EVENT - A basic initiating fault requiring no further development
	CONDITIONING EVENT - Specific conditions or restrictions that apply to any logic gate (used primarily with PRIORITY AND and INHIBIT gates)
	UNDEVELOPED EVENT - An event which is not further developed either because it is of insufficient consequence or because information is unavailable
	HOUSE EVENT - An event which is normally expected to occur
GATE SYMBOLS	
	AND - Output fault occurs if all of the input faults occur
	OR - Output fault occurs if a least one of the input faults occurs
	COMBINATION - Output fault occurs if n of the input faults occur
	EXCLUSIVE OR - Output fault occurs if exactly one of the input faults occurs
	PRIORITY AND - Output fault occurs if all of the input faults occur in a specific sequence (the sequence is represented by a CONDITIONING EVENT drawn to the right of the gate)
	INHIBIT - Output fault occurs if the (single) input fault occurs in the presence of an enabling condition (the enabling condition is represented by a CONDTIONING EVENT drawn to the right of the gate)
TRANSFER SYMBOLS	
	TRANSFER IN - Indicates that the tree is developed further at the occurrence of the corresponding TRANSFER OUT (e.g., on another page)
	TRANSFER OUT - Indicates that this portion of the tree must be attached at the corresponding TRANSFER IN

Gates

There are two basic types of fault tree gates, the OR-gate and the AND-gate. All other gates (other than the special DFT gates described in Chapter 8) are special cases of these two basic types. With one exception, gates are symbolized by a shield with a flat or curved base.

The OR-Gate



The OR-gate is used to show that the output event occurs only if one or more of the input events occur. There may be any number of input events to an OR-gate.

Figure 4-2 shows a typical two-input OR-gate with input events A and B and output event Q. Event Q occurs if A occurs, B occurs, or both A and B occur.

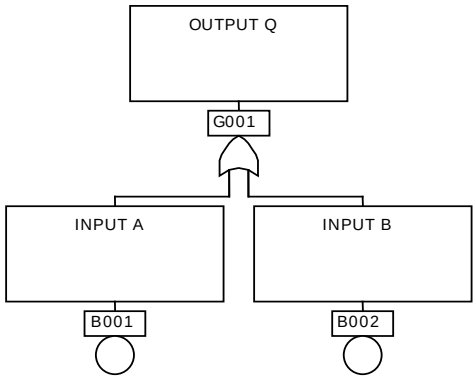


Figure 4-2. The OR-Gate

Inputs to an OR-gate are restatements of the output but are more specifically defined as to cause or to specific scenario. Figure 4-3 helps to clarify this point.

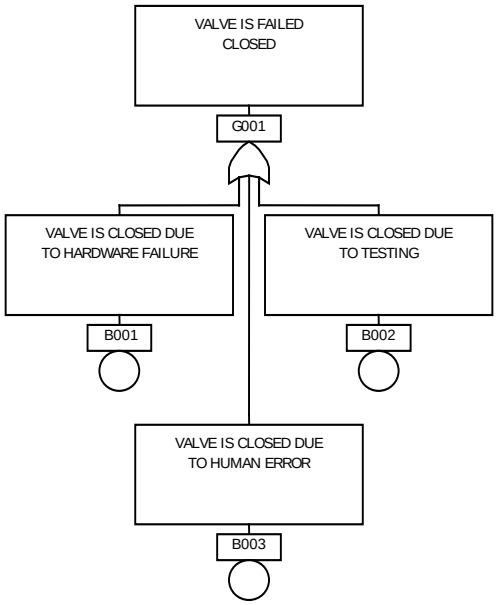


Figure 4-3. Specific Example of the OR-Gate.

Note that the subevents in Figure 4-3 can be further developed; for instance, see Figure 4-4.

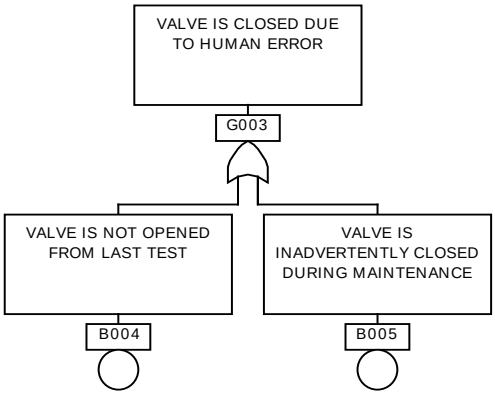
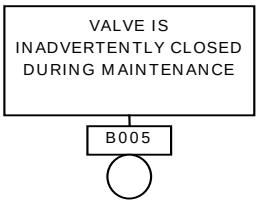
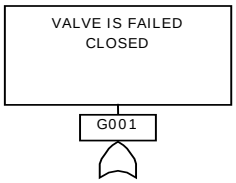


Figure 4-4. OR-Gate for Human Error

However, the event

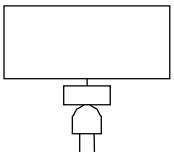


is still a restatement of the output event of the first OR-gate



with regard to a specific cause.

The AND-Gate



The AND-gate is used to show that the output fault occurs only if all the input faults occur. There may be any number of input faults to an AND-gate. Figure 4-5 shows a typical two-input AND-gate with input events Input A and Input B, and output event Output Q. Output Q occurs only if Input A and Input B both occur.

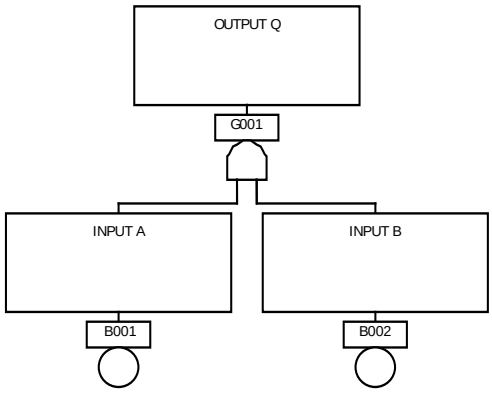


Figure 4-5. The AND-Gate

In contrast to the OR-gate, causes can directly input an AND-gate; that is, the input faults collectively represent the cause of the output fault. An example of an AND-gate is shown in Figure 4-6. A failure of both fuel cells and of the battery will result in a failure of all power to the DC bus.

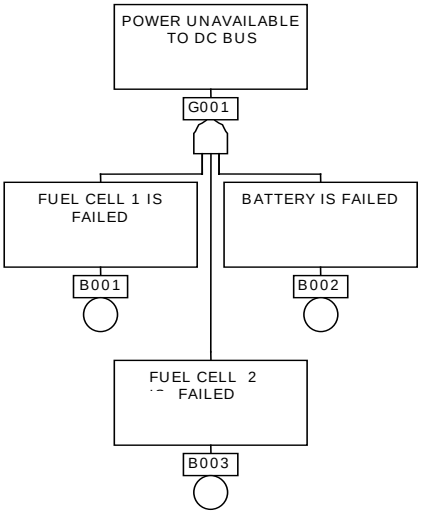


Figure 4-6. Specific Example of an AND-Gate

When describing the events input to an AND-gate, any dependencies must be incorporated in the event definitions if the dependencies affect the system logic. Dependencies generally exist when the failure "changes" the system. For example, when the first failure occurs (e.g., Input A of Figure 4-5), the system may automatically switch in a standby unit. The second failure, Input B of Figure 4-5, is now analyzed with the standby unit assumed to be in place. In this case, Input B of Figure 4-5 would be more precisely defined as "Input B given the occurrence of A."

The variant of the AND-gate shown in Figure 4-7 explicitly shows such dependencies and is useful for those situations when the occurrence of one of the faults alters the operating modes and/or stress levels in the system in a manner affecting the occurrence mechanism of the other fault.

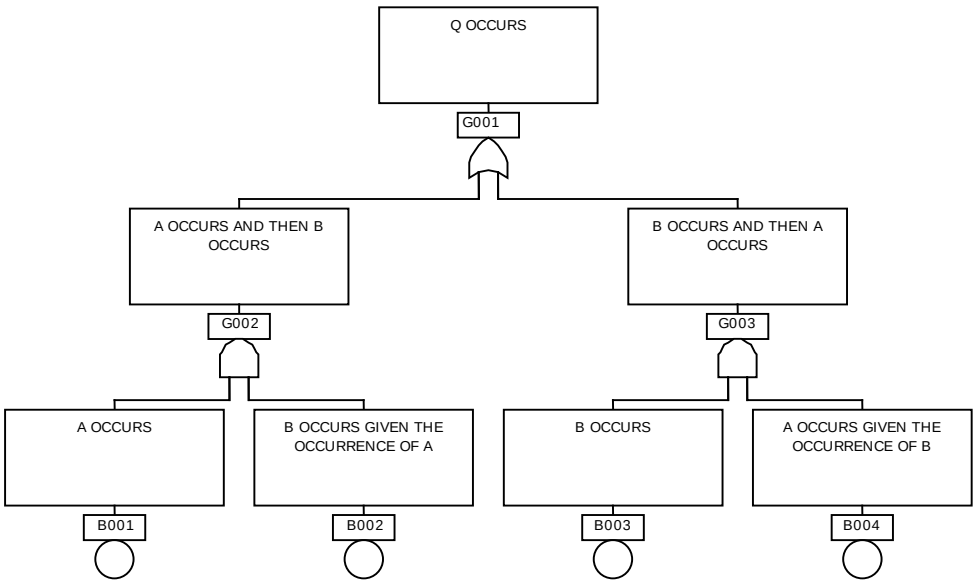
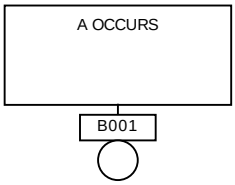
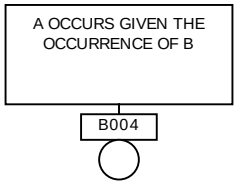


Figure 4-7. AND-Gate Relationship with Dependency Explicitly Shown

That is, the subtree describing the mechanisms or antecedent causes of the event

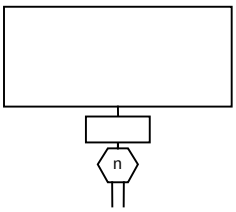


will be different from the subtree describing the mechanisms for the event.



For multiple inputs to an AND-gate with dependencies affecting system logic among the input events, the "givens" must incorporate all preceding events.

The COMBINATION-Gate



The COMBINATION-gate, represented by a hexagon with a number at its center, is a modeling convenience available in most software packages. The combination gate allows a user to specify the number of failures within a group of inputs that will result in output from the gate. This gate eliminates the need for an analyst to delineate all the required combinations of input events that can cause in the output event for cases when fewer than the total number of inputs are required (e.g., three of four inputs).

An example of the use of the COMBINATION-gate is shown in Figures 4-8 and 4-9. Figure 4-8 is a model, using only OR- and AND-gates, of a system in which the failure of two of three trains will result in system failure. As can be seen, each combination of train failure must be included in the FT. In Figure 4-9, the same failure logic is represented using the COMBINATION-gate. In addition to modeling convenience, the COMBINATION-gate helps eliminate errors that can result from the specification of multiple failure combinations.

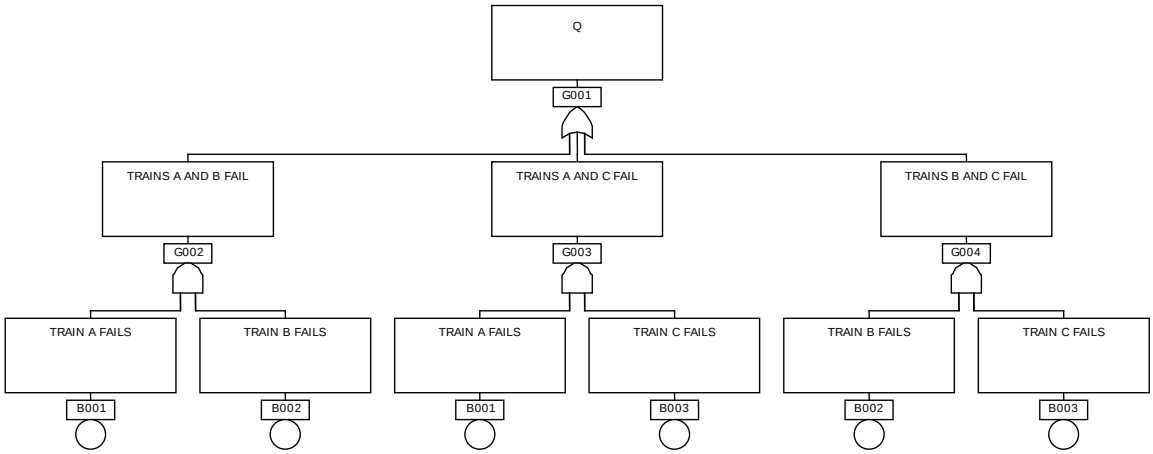


Figure 4-8. Example of the COMBINATION-Gate (AND- and OR-Gate Model).

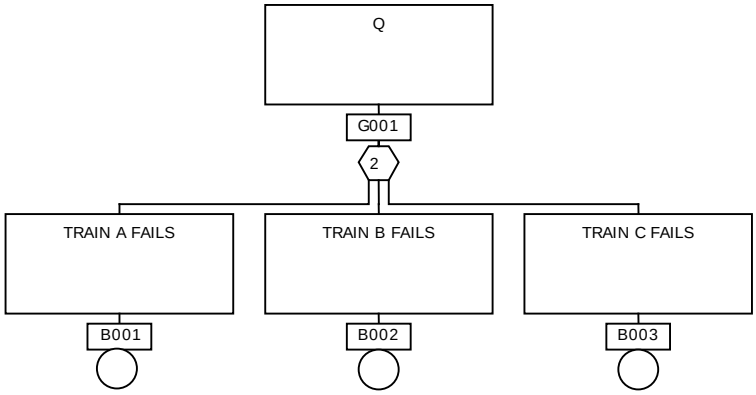
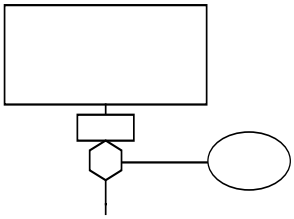


Figure 4-9. Example of the COMBINATION-Gate.

The INHIBIT-Gate



The INHIBIT-gate, represented by the hexagon, is a special case of the AND-gate. The output is caused by a single input, but some qualifying condition must be satisfied before the input can produce the output. The condition that must exist is the conditional input. A description of this conditional input is spelled out within an elliptic event drawn to the right of the gate. Figure 4-10 shows a typical INHIBIT-gate with Input A, Conditional Input B and Output Q. Event Q occurs only if Input A occurs under the condition specified by Input B.

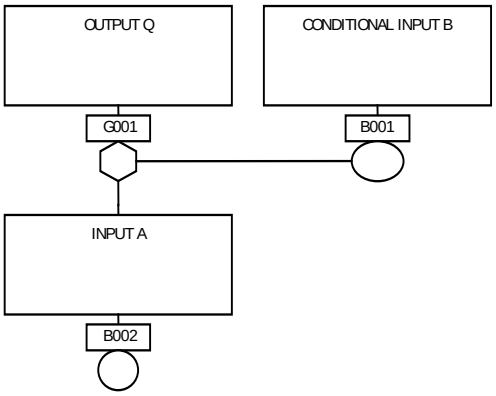


Figure 4-10. The INHIBIT-Gate

To clarify this concept, two examples are given below and are illustrated in Figure 4-11.

(a) Many chemical reactions go to completion only in the presence of a catalyst. The catalyst does not participate in the reaction, but its presence is necessary.

(b) If the loss of resiliency in an O-ring constitutes an event of interest, such an event can occur only when the temperature T is less than T(critical), the temperature at which the material of which the O-ring is made is no longer pliable. In this case the output event would be "O-ring Failure," the input event would be "Existence of Low Temperature," and the conditional input would be "T < T(critical)."

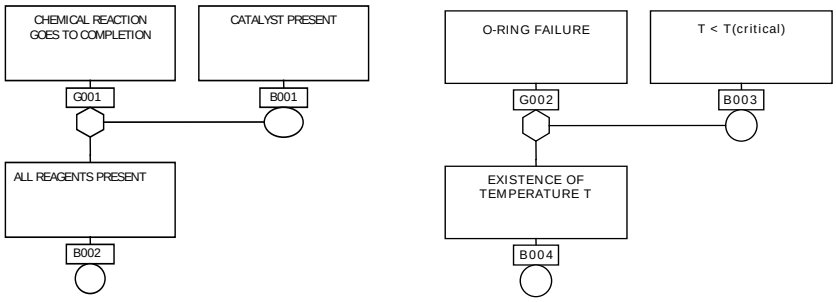


Figure 4-11. Examples of the INHIBIT-Gate

Occasionally, especially in the investigation of secondary failures**,** another type of INHIBIT-gate depicted in Figure 4-12 is used.

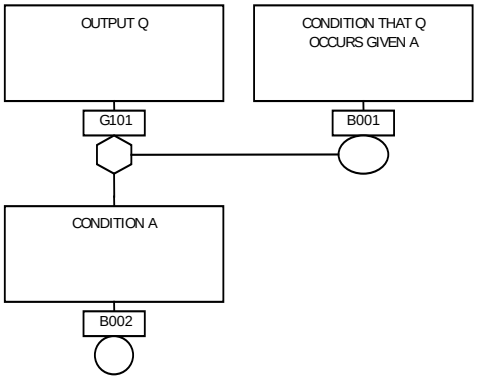
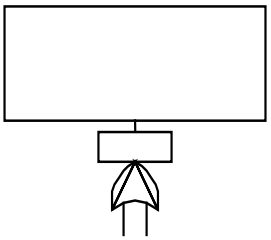


Figure 4-12. An Alternative Type of INHIBIT-Gate

In Figure 4-12, condition A is the necessary, but not always sufficient, single cause of output Q; i.e., for Q to occur we must have A, but just because A occurs it does not mean that Q follows inevitably. The portion of time Q occurs when A occurs is given in the conditional input ellipse.

The gates we have described above are the ones most commonly used and are now standard in the field of fault tree analysis. However, a few other special purpose gates are sometimes encountered.

The EXCLUSIVE OR-gate



The EXCLUSIVE OR-gate is a special case of the OR-gate. In most FT modeling, the EXCLUSIVE OR-gate is a two-input gate in which the output event occurs only if one of the inputs occurs but not two[1]. Figure 4-13 depicts a typical EXCLUSIVE OR-gate.

[1] Some FT modeling approaches recognize an EXCLUSIVE OR-gate with more than two inputs. In this situation, the logic represented by the gate can be dependent on the application software and its intended use. In an application associated with electronic logic circuits, for example, the output of an EXCLUSIVE OR-gate will occur when an odd number of inputs occur (i.e., three inputs to a three-input gate).

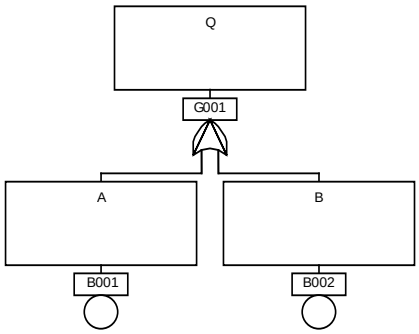
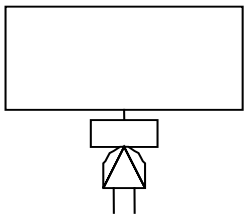


Figure 4-13. The Exclusive OR-Gate

The exclusive OR differs from the usual or inclusive OR in that the situation where both input events occur is precluded. Thus, the output event Q occurs if A occurs or B occurs, but not if both A and B occur. As will be seen later, the quantitative difference between the inclusive and exclusive OR-gates is generally so insignificant that the distinction is not usually necessary. In those special instances where the distinction is significant, this difference can be accounted for in the quantification phase.

The PRIORITY AND-gate



The PRIORITY AND-gate is a special case of the AND-gate in which the output event occurs only if all input events occur in a specified ordered sequence. The sequence is usually shown inside an ellipse drawn to the right of the gate. Figure 4-14 shows two alternative ways of depicting a typical PRIORITY AND-gate with two inputs.

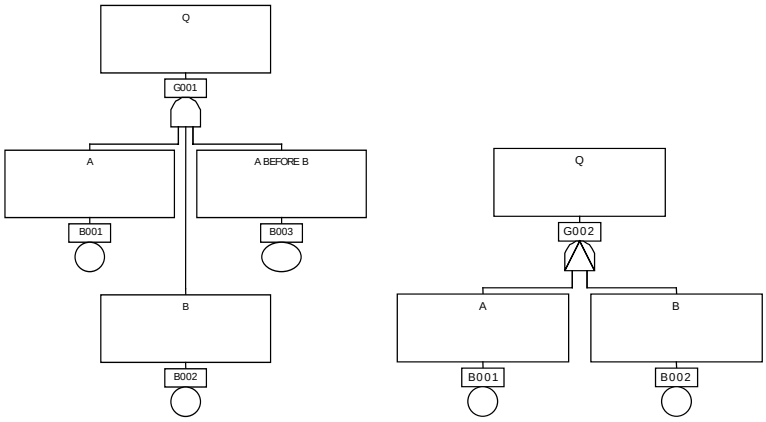
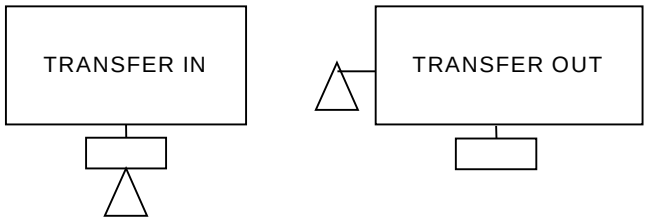


Figure 4-14. The PRIORITY AND-Gate

In Figure 4-14, the output event Q occurs only if both input events A and B occur with A occurring before B.

Transfer Symbols

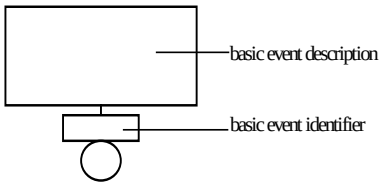


The triangles are introduced as transfer symbols and are used as a matter of convenience to avoid extensive duplication in a fault tree or to allow a large tree to be represented on a number or smaller trees for clarity. A "transfer in" gate will link to its corresponding "transfer out." This "transfer out," perhaps on another sheet of paper, will contain a further portion of the tree describing input to the gate.

Basic Events

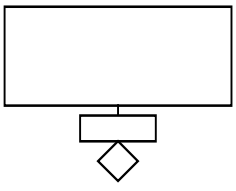
The basic events of a fault tree are those events, which, for one reason or another, have not been further developed. These are the events for which probabilities will have to be provided if the fault tree is to be used for computing the probability of the top event. There are four types of basic events. These are:

The Primary Event



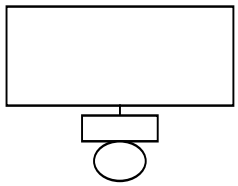
The circle describes a basic initiating fault event that requires no further development. In other words, the circle signifies that the appropriate limit of resolution has been reached.

The Undeveloped Event



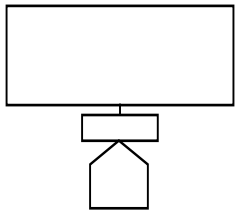
The diamond describes a specific fault event that is not further developed, either because the event is of insufficient consequence or because information relevant to the event is unavailable.

The Conditioning Event



The ellipse is used to record any conditions or restrictions that apply to any logic gate. It is used primarily with the *INHIBIT* and *PRIORITY AND*-gates.

The House Event



The house is used to signify an event that is normally expected to occur: e.g., a phase change in a dynamic system. Thus, the house symbol displays events that are not, of themselves, faults.

4.2 Component Fault Categories: Primary, Secondary, and Command

It is useful for the fault tree analyst to classify faults into three categories: primary, secondary and command. A primary fault is any fault of a component that occurs in an environment for which the component is qualified; e.g., a pressure tank, designed to withstand pressures up to and including a pressure P_O, ruptures at some pressure $p \leq P_O$ because of a defective weld.

A secondary fault is any fault of a component that occurs in an environment for which it has not been qualified. In other words, the component fails in a situation which exceeds the conditions for which it was designed; e.g., a pressure tank, designed to withstand pressure up to and including a pressure P_O, ruptures under a pressure $p>P_O$. Other types of secondary faults include damage from micrometeoroid impact, failure due to loss of environmental control systems or excessive vibration.

Because primary and secondary faults are generally component failures, they are usually called primary and secondary failures. A command fault in contrast, involves the proper operation of a component but at the wrong time or in the wrong place; e.g., an arming device in a warhead closes too soon because of a premature or otherwise erroneous signal originating from some upstream device. Another type of command fault is the failure of a component to operate due to a loss of signal or control power.

Oftentimes, the analyst doesn't think of these fault categories when constructing a FT. Secondary failures may, for example, not be included under a tacit assumption that only primary failures and design conditions are considered. However, there has been many a FT constructed

that inadvertently omitted one or more of these faults due to shortcut thinking. Even if the analyst doesn't explicitly use these categories, they serve as a useful checklist to assure that the FT is complete in its coverage of the different types of faults.

4.3 Passive vs. Active Components

In most cases it is convenient to separate components into two types, passive and active (also called quasi-static and dynamic). A passive component contributes in a more-or-less static manner to the functioning of the system. Such a component may act as a transmitter of energy from place to place (e.g., a wire or bus-bar carrying current or a steam line transmitting heat energy), or it may act as a transmitter of loads (e.g., a structural member). To assess the operation of a passive component, tests such as stress analysis, heat transfer studies, etc. are performed. Further examples of passive components are pipes, bearings, journals, and welds.

An active component contributes in a more dynamic manner to the functioning of its parent system by modifying system behavior in some way. Generally, such a component requires an input signal or trigger for its output signal. In such cases the active component acts as a "transfer function," a term widely used in electrical and mathematical studies. If an active component fails, there may be no output signal or there may be an incorrect output signal. A valve that opens and closes, for example, modifies the system's fluid flow, and a switch has a similar effect on the current in an electrical circuit. To assess the operation of an active component, parametric studies of operating characteristics and studies of functional interrelationships are performed.

A passive component can be considered as the transmitter of a "signal." The physical nature of this "signal" may vary considerably; for example, it may be a current or force. A passive component may also be thought of as the "mechanism" (e.g., a pipe) whereby the output of one active component becomes the input to a second active component. The failure of a passive component will result in the loss of transmission of its "signal."

As an example, consider a postman (passive component) who transmits a signal (letter) from one active component (sender) to another (receiver). The receiver will then respond in some way (provide an output) as a result of the message (input) that has been received.

From a numerical reliability standpoint, an important difference between failures of active components and failures of passive components is the difference in failure rates. The difference in reliability between the two types of components is, quite commonly, two to three orders of magnitude.

In the above discussion, the definitions of active and passive components apply to the primary function performed by the component: failures of the component related to the failure of that primary function. (For example, "passive" failure modes of active components, e.g., valve rupture, might be considered if we attempted to classify specific failure modes according to the "active" or "passive" definition.)

4.4 The "Immediate Cause" Concept

Returning to the perspective of the system analyst, the system under study is first defined (its boundary is determined) and a particular system failure mode is selected for further analysis.

The latter constitutes the top event of the fault tree. The analyst next determines the *immediate, necessary, and sufficient* causes for the occurrence of this top event. It should be noted that these are not the basic causes of the event but the *immediate* causes or *immediate* mechanisms for the event. This is an extremely important point that will be clarified and illustrated in later examples.

The immediate, necessary, and sufficient causes of the top event are now treated as sub-top events and the analyst then proceeds to determine their immediate, necessary, and sufficient causes. In so doing, the analyst is placed in the position of the subsystems analyst for whom the failure mechanisms are the failure modes; that is, the sub-top events correspond to the top events in the subsystem fault tree.

In this way the analyst proceeds down the tree continually transferring the point of view from mechanism to mode, and continually approaching finer resolution in defining mechanisms and modes, until ultimately, the limit of resolution of the tree is reached. This limit consists of basic component failures of one sort or another, and the tree is now complete.

As an example of the application of the "immediate cause" concept, consider the simple system in Figure 4-15.

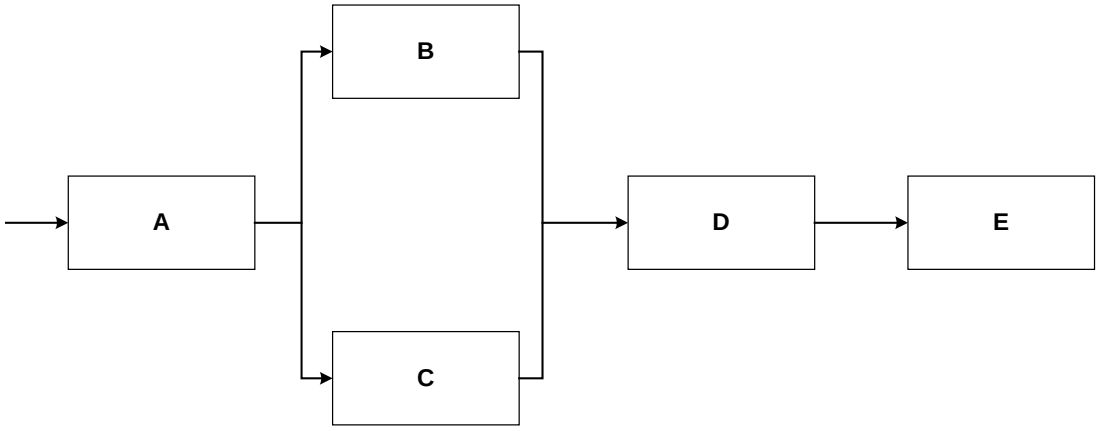


Figure 4-15. System Illustrating "Immediate Cause" Concept

This system is intended to operate in the following way: a signal to A triggers an output from A which provides inputs to B and C. B and C then pass a signal to D which finally passes a signal to E. A, B, C and D are dynamic subsystems. Furthermore, subsystem D needs an input signal from either B or C or both to trigger its output to E. Thus there is redundancy in this portion of the system.

The system of Figure 4-13 can be interpreted quite generally. For example, it could represent an electrical system in which the subsystems are analog modules (e.g., comparators, amplifiers) it could be a piping system in which A, B, C and D are valves; or it could represent a portion of the "chain of command" in a corporation.

Consider as the top event the possible outcome "no signal to E," and assume that the transmitting devices (passive components) which pass the signals from one subsystem to another can be

neglected. This is tantamount to assigning a zero failure probability to the wires, pipes, or command links.

A step-by-step analysis of the top event is begun. The immediate cause of the event, "no signal to E," is "no output from D." The analyst should strongly resist the temptation to list the event, "no input to D" as the immediate cause of "no signal to E." In the determination of immediate causes, one step should be taken at a time. The "immediate cause" concept is sometimes called the "Think Small" Rule because of the methodical, one-step-at-a-time approach.

Now the sub-top event, "no output from D," has been identified and it is next necessary to determine its immediate cause or causes. There are two possibilities:

(1) "There is an input to D but no output from D."
(2) "There is no input to D."

Therefore, the sub-top event, "no output from D," can arise from the union of events 1 or 2. (The reader should note that if we had taken more than one step and had identified (improperly) the cause of "no input to D," then event 1 above would have been missed. In fact, the motivation for considering immediate causes is now clear: it provides assurance that no fault event in the sequence is overlooked.)

Now the immediate causes for the new mode failures, events 1 and 2 can be investigated. If the limit of resolution is the subsystem level, then event 1 (which can be rephrased, "D fails to perform its proper function due to some fault internal to D") is not analyzed further and constitutes a basic input to the tree. With respect to event 2, its immediate, necessary and sufficient cause is "no output from B and no output from C," which appears as an intersection of two events, i.e.,

2 = 3 and 4

where

3 = "no output from B," and
4 = "no output from C."

As a matter of terminology, it is convenient to refer to events as "faults" if they are analyzed further (e.g., event 2). However, an event such as 1 which represents a basic tree input and is not analyzed further is referred to as a "failure." This terminology is also fairly consistent with the mechanistic definitions of "fault" and "failure" given previously.

The analysis is continued by focusing attention on events 3 and 4. As far as 3 is concerned,

3 = 5 or 6

where

5 = "input to B but no output from B," and
6 = "no input to B."

Event 5 can be readily identified as a failure (basic tree input). Event 6 is a fault which can be analyzed further. Event 4 is dealt with in an analogous way.

The further steps in the analysis of this system can now be easily supplied by the reader. The analysis will be terminated when all the relevant basic tree inputs have been identified. In this regard, the event "no input to A" is also considered to be a basic tree input.

Our analysis of the top event (“no input to E”) consequently produced a linkage of fault events connected by “and” and “or” logic. The framework (or system model) on which this linkage is “hung” is the fault tree. The next section provides the necessary details for connecting the fault event linkage to its framework (fault tree).

4.5 Basic Rules for Fault Tree Construction

The construction of fault trees is a process that has evolved gradually over a period of about 50 years. In the beginning it was thought of as an art, but it was soon realized that successful trees were best drawn in accordance with a set of basic rules. Observance of these rules helps to ensure successful fault trees so that the process is now less of an art and more of a science. The basic rules for successful fault tree analysis will now be examined.

Consider Figure 4-16. It is a simple fault tree or perhaps a part of a larger fault tree. Note that none of the failure events have been “written in”; they have been designated Q, A, B, C, and D.

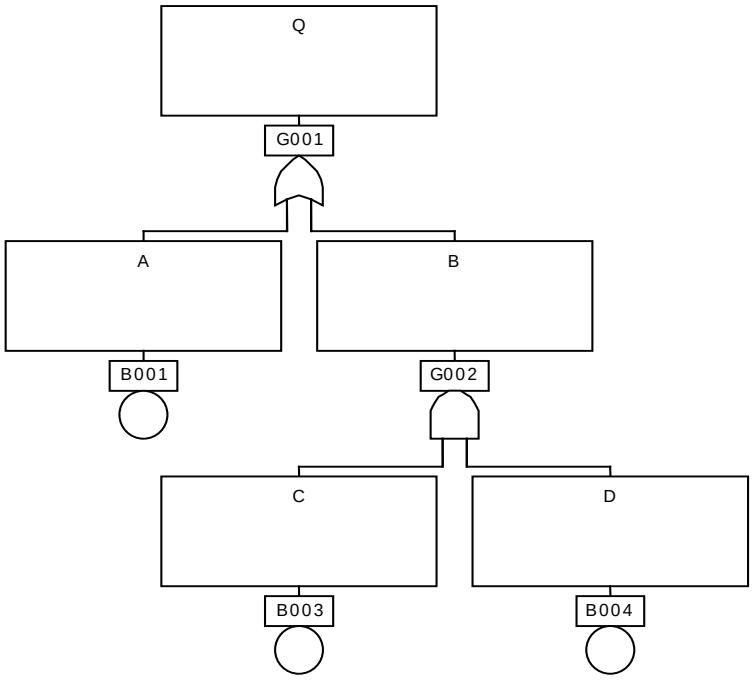


Figure 4-16. A Simple Fault Tree

Now, when a specific problem is considered it is necessary to describe exactly what such events such as Q, A, B, C, D actually are. The proper procedure for doing this constitutes **Ground Rule 1:**

> **Write the statements that are entered in the event boxes as faults; state precisely what the fault is and the conditions under which it occurs. Do not mix successes with faults.**

The "what-condition" describes the relevant failed (or operating) state of the component. The "when-condition" describes the condition of the system—with respect to the component of interest—that makes that particular state of existence of the component a fault.

Note that Ground Rule 1 may frequently require a fairly verbose statement. If this is the case, so be it. The analyst is cautioned not to be afraid of wordy statements. Do not tailor the length of your statement to the size of the box that you have drawn. If necessary, make the box bigger or the font size smaller! (Modern FT software may limit the length of a statement. However, allowable description lengths are usually sufficient to adequately describe an event.) It is permissible to abbreviate words but resist the temptation to abbreviate ideas. Examples of fault statements are:

(1) Normally closed relay contacts fail to open when EMF is applied to coil.
(2) Motor fails to start when power is applied.

The next step in the procedure is to determine the necessary and sufficient events that result in the fault described by each boxed statement. A useful step in determining the next logic structure to develop is to ask whether the fault is a "state of component" fault or whether it is a "state of system" fault. A "state of component" fault is a fault that is localized to a given component. A "state of system" fault is a fault that is not necessarily localized to a given component, but may involve a system-level fault or the occurrence of multiple faults.

The "state of component" versus "state of system" fault terminology does not necessarily have to be used by the analyst. However it provides a useful checklist or mnemonic. What is important is the questioning of whether the fault is localized to a component or whether it can involve system-level faults or multiple faults.

Using "state of component" and "state of system" as mnemonics leads to **Ground Rule II:**

> **If the answer to the question, "Is this fault a component failure?" is "Yes," classify the event as a "state of component fault." If the answer is "No," classify the event as a "state of system fault."**

If the fault event is classified as "state of component," add an OR-gate below the event and look for primary, secondary and command failure modes. If the fault event is classified as "state of system," look for the minimum necessary and sufficient immediate cause or causes. A "state of system" fault event may require an AND-gate, an OR-gate, an INHIBIT-gate, or possibly no gate at all. As a general rule, when energy originates from a point outside the component, the event may be classified as "state of system."

To illustrate Ground Rule II, consider the simple motor-switch-battery circuit depicted in Figure 4-17.

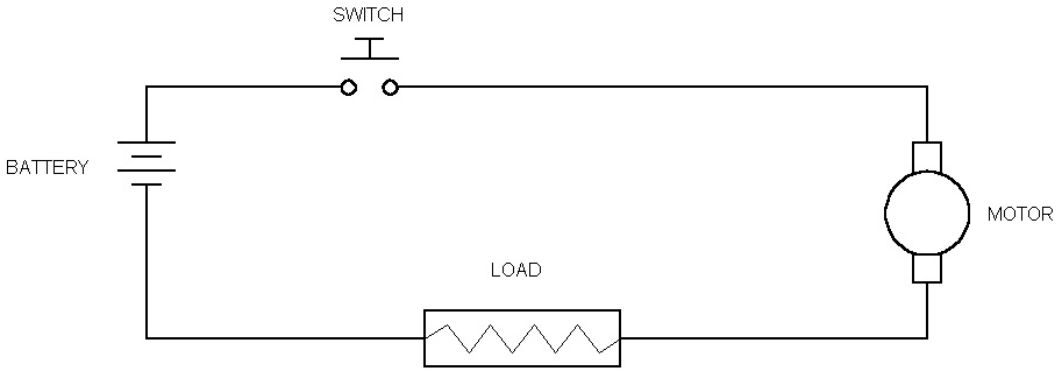


Figure 4-17. Simple Motor-Switch-Battery System

The system can exist in two states: operating and standby. The following faults can be identified and classified using Ground Rule II:

OPERATING STATE

FAULT	CLASSIFICATION
Switch fails to close when thumb pressure is applied.	State of component
Switch inadvertently opens when thumb pressure is applied	State of component
Motor fails to start when power is applied to its terminals.	State of component
Motor ceases to run with power applied to terminals	State of component

STANDBY STATE

FAULT	CLASSIFICATION
Switch inadvertently closes with no thumb pressure applied.	State of component
Motor inadvertently starts.	State of system

In addition to the above ground rules, there are a number of other procedural statements that have been developed over the years. The first of these is the **No Miracles Rule:**

> **If the normal functioning of a component propagates a fault sequence, then it is assumed that the component functions normally.**

It might be discovered in the course of a system analysis that the propagation of a particular fault sequence could be blocked by the miraculous and totally unexpected failure of some component. The correct assumption to make is that the component functions normally, thus allowing the passage of the fault sequence in question. However, if the normal functioning of a component acts to block the propagation of a fault sequence, then that normal functioning must be defeated by faults if the fault sequence is to continue up the tree. Another way of stating this is to say that, if an AND situation exists in the system, the model must take it into account.

Two other procedural statements address the dangers of not being methodical and attempting to shortcut the analysis process. The first is the **Complete-the-Gate Rule:**

> **All inputs to a particular gate should be completely defined before further analysis of any one of them is undertaken.**

The second is the **No Gate-to-Gate Rule:**

> **Gate inputs should be properly defined fault events, and gates should not be directly connected to other gates.**

The Complete-the-Gate Rule states that the fault tree should be developed in levels, and each level should be completed before any consideration is given to a lower level. Concerning the No-Gate-to-Gate Rule, a "shortcut" fault tree is shown in Figure 4-18. Note that current FT software often includes a description box as part of the gate object, which facilitates a clear development of the fault tree logic and prevents the gate-to-gate connections shown in Figure 4-18.

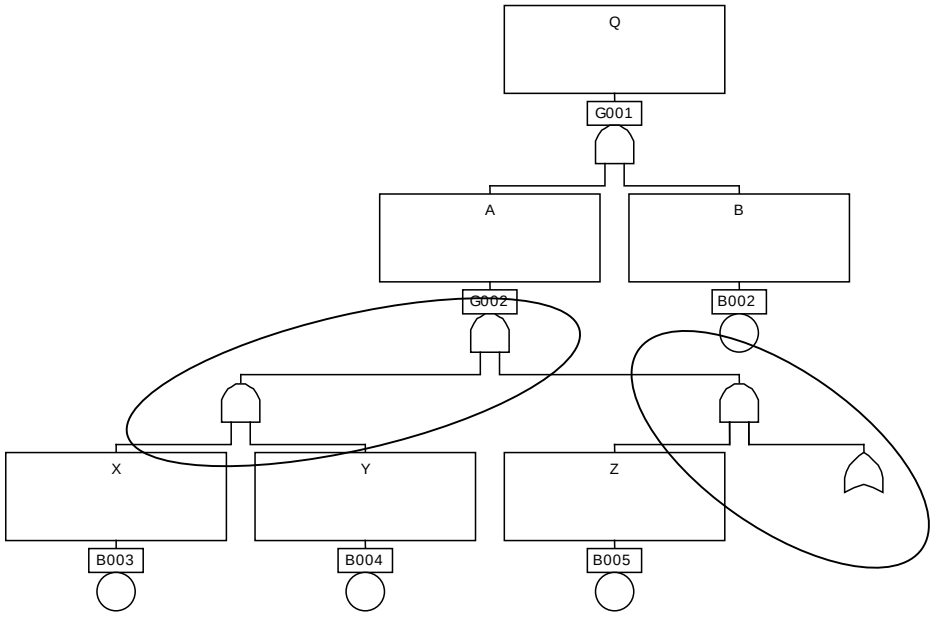


Figure 4-18. A Short-Cut Fault Tree

The "gate-to-gate" connection is indicative of sloppy analysis. When a fault tree is being constructed, the gate-to-gate shortcuts may lead to confusion and may demonstrate that the analyst has an incomplete understanding of the system. A fault tree can be successful only if the analyst has a clear and complete understanding of the system to be modeled.

4.6 State of System Versus State of Component Faults

As indicated, as a mnemonic it is useful to ask whether a fault is a "state of system" fault or a "state of component" fault. A "state of component" fault is a fault is associated uniquely with one component. A "state of system" fault is a fault not uniquely associated with one component. The immediate cause of a "state of system" fault involves more than one component. An example of a "state of component" fault is a valve failure to open. The valve may fail to open because of an inherent valve failure or because it did not receive an open signal. Both these faults uniquely involve the valve.

An example of a "state of system" fault is no flow from a redundant pair of pumps. The faults involve two pumps. The analysis may find that a primary cause is no water supply to the pumps from a single tank. However, the immediate cause of no water supply is no supply from either of two pumps. Another example of a "state of system" fault is a pump inadvertently starting. The pump did not have the fault. It was the inadvertent application of power to the pump that was the fault. Therefore it is a "state of system" fault.

A "state of component" fault is always modeled as an OR gate or a primary event. The example above of the valve failing to open is modeled as an OR gate with the primary event of valve failure as one input and failure of the signal to the valve as the other input. An example of a "state of component" fault that is a primary event "valve failed closed." There is no signal to the valve and the normal position of the valve is open. The valve failing closed could be due to an internal blockage or failure, for example.

A "state of system" fault is modeled with the type of gate that is applicable. For example, for the redundant pump example above, if both pumps are needed then the gate is an OR gate. If either pump is sufficient then the "state of system" fault is modeled with an AND gate.

4.7 Depth to Which a Fault Tree Should be Constructed

Even though this topic is included in other discussions it is useful to highlight it here. The depth to which a fault tree is constructed is important since it determines how meaningful the end result is. There have been more cases of the fault trees being developed to too great a depth than not enough depth. When the fault tree is developed to too great a depth then not only is this a waste of effort but it introduces additional uncertainties and obfuscation into the analysis.

The general principal is that the fault tree should be developed to the necessary depth to identify functional dependencies and to a depth that is consistent with the data available and the objectives of the analysis. Developing the fault tree to greater depth loses cohesiveness and structure. For example, a valve can be decomposed into over one hundred parts. Each part would show up in the results. It would be difficult to synthesize these part results to recognize the valve as the entity. Furthermore, data is extremely sparse or does not exist at such deep levels of definition so that quantification is impossible or at best bloated with uncertainty.

Usually a fault tree of a system failure is developed to a major component or contributor level. Examples of major components and contributors are valves, pumps, identifiable human errors, and fire initiations. It is important to show the support interfaces such as power supplies and cooling water supplies. These interfaces are what determine whether there are any hardwired dependencies or functional dependencies among the components or major contributors. Developing depth for depth sake is a misguided effort. However, developing a fault tree to the depth necessary to identify dependencies and relationships among events is essential.

4.8 FMEAs Cannot be Combined to Make a Fault Tree

There have been attempts to combine FMEAs to form a fault tree. This cannot be done and is bad practice. An FMEA connects given initiating causes to their end results or consequences. These consequences are often failure of a system or component. Because the consequences are the same as those that can appear in a fault tree, it might appear as if these consequences can be linked, or be "patched," to form a top event or an intermediate event in a fault tree. This cannot be done and will produce an erroneous model.

The reason that this patching cannot be done is because a fault tree is a top down structured model of the immediate, necessary and sufficient relationships among events that result in the top event. The basic events are identified in an exhaustive manner that fit into this relationship. Patching together FMEAs developed using a bottom up approach does not constitute a fault tree analysis. There is no assurance, as in a fault tree, that the primary causes of the FMEAs have been identified in a comprehensive and consistent manner as in a fault tree. This is because in an FMEA the primary causes are selected without a defining logic structure. Moreover, an FMEA does not show the relationships among the primary causes and their subsequent events. These relationships can result in causes or events being redundant to others or being contained (absorbed) in others. These relationships are accurately identified in a fault tree that is used to determine the resulting basic events that are consistent and complete within this framework.

This is not meant to imply that an FMEA cannot be a useful tool in the construction of a fault tree. In particular, an FMEA can be used to check a fault tree. If the root causes (or hazardous initiating events) initially identified in an FMEA are systematically developed through FTA (i.e., through the use of an MLD), then the initial use of FMEA can help provide assurance of the completeness of a given FT analysis objective. An FMEA can also serve as a tool for initial evaluation to assist in deciding whether a fault tree need be constructed and to what detail. However, patching the results of one or more FMEAs together to construct an FT is piecemeal and does not produce a meaningful or useful model.

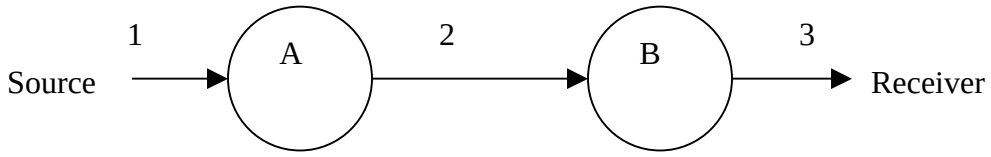
5. Extended FTA

The following sections describe FTA topics that are relevant in applications and implementations. The first topic discusses the basic fault tree construction principle as applied to systems in which fluid flows or to networks in which signals or power flows. Common cause failure (CCF) modeling, human error modeling, modeling loops and feedback, modeling phenomenological sequences, naming schemes for a fault tree, construction ground rules that have been used in FTA, and methods available for validating a fault tree are then discussed.

5.1 Modeling Inadvertent Flow Versus No Flow: An Illustration of the Basic Fault Tree Modeling Principle

The basic principle of fault tree modeling is to identify in each step the immediate, necessary, and sufficient causes of the fault being analyzed. To better see how this principle is explicitly applied, consider the modeling of inadvertent flow versus the modeling of no flow. The flow can be a liquid flow, gaseous flow, or a general signal flow. The example is not only useful in showing how the fault tree principle is applied but also to provide a basic modeling template that can be applied in many actual fault tree models.

Consider the signal line below. Treat A and B as if they were valves and the fluid flowing as water.



The Source will then be the supply of water and the Receiver (or Receptacle) the receiving component. Alternatively, consider a signal flow from the Source to the Receiver with Components A and B as relays or other electrical components. The numbers in the figure identify the flow lines.

The fault tree logic and models for no flow to the receiver (Figure 5-1) and for inadvertent flow to the receiver (Figure 5-2) are given below. The reader should be able to identify a one-to-one relationship between each step of the thought process logic and the corresponding step in the tree. The equal sign in the logic equations is logical equality and can be read as "is equivalent to" or "results from". The analysis proceeds in stepwise fashion by backwards tracing the no flow (or inadvertent flow) in a deductive manner from the receiver to the source. For each step the question concerning what are the immediate, necessary, and sufficient conditions that result in the event is asked. Finally, note the parallelism in the two models with AND gates replacing OR gates for the inadvertent flow. This parallelism generally occurs.

Logic for No Flow to Receiver

1. No Flow to Receiver = No Flow in Line 3
2. No Flow in Line 3 = Component B Blocks Flow OR No Flow in Line 2
3. No Flow in Line 2 = Component A Blocks Flow OR No Flow in Line 1
4. No Flow in Line 1 = No Flow from Source

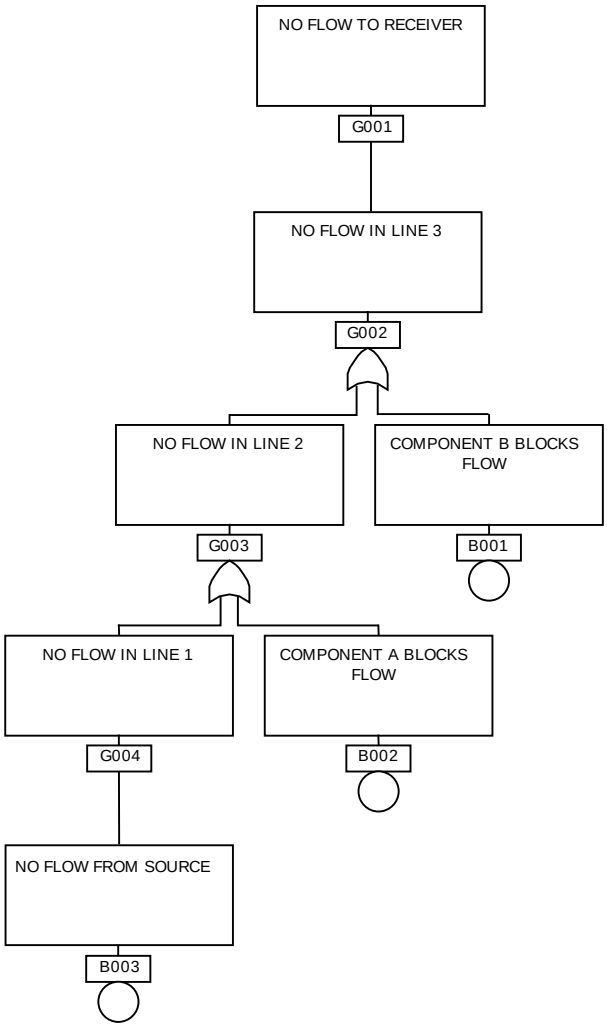


Figure 5-1. Fault Tree Model for No Flow to Receiver.

Logic for Inadvertent Flow to Receiver

1. Inadvertent Flow to Receiver = Inadvertent Flow in Line 3
2. Inadvertent Flow in Line 3 = Component B Passes Flow AND Inadvertent Flow in Line 2
3. Inadvertent Flow in Line 2 = Component A Passes Flow AND Inadvertent Flow in Line 1
4. Inadvertent Flow in Line 1 = Inadvertent Flow from Source

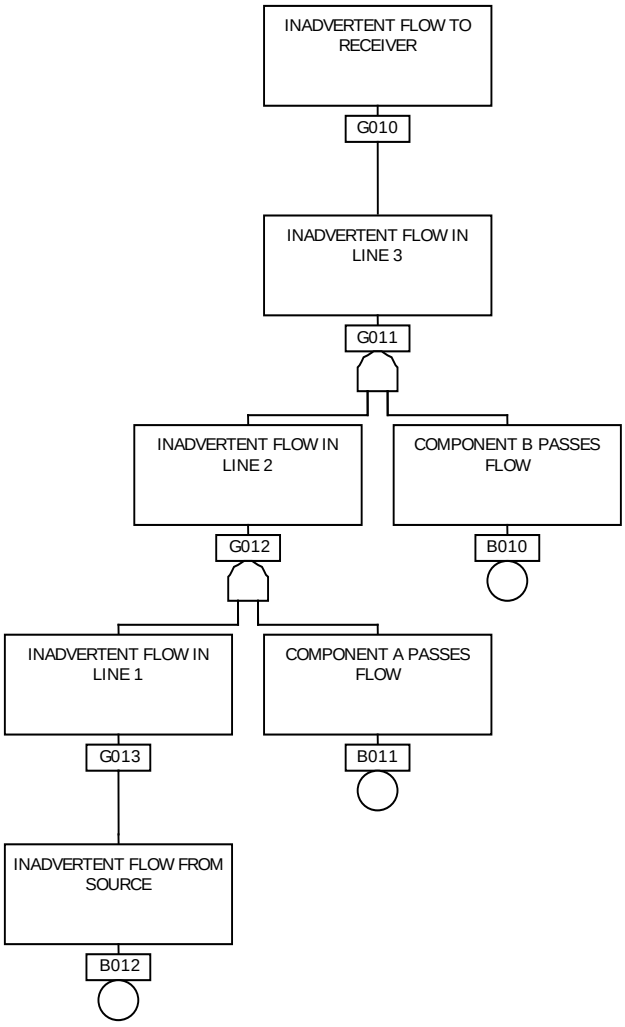


Figure 5-2. Fault Tree Model for Inadvertent Flow to Receiver.

A comparison of the FTs in Figures 5-1 and 5-2 illustrates the importance of the top event definition in establishing the structure of the subsequent fault tree. As shown in the fault tree in Figure 5-2, multiple failures must occur to result in inadvertent flow to the receiver. However, no flow to the receiver (Figure 5-1) can result from any one of three failures. The applicable component failure modes are also different in the two FTs, which would require different data if the trees were to be quantified.

5.2 Modeling of Common Cause Failures in a Fault Tree

Common cause failures (CCFs) are two or more component failures that occur at the same time or occur within a relatively short interval and that are due to a common cause. "Component" is used in a general sense here and can mean a component such as a valve or pump or a subsystem such as a power supply. The common causes of the failures that are considered in a fault tree are ones which are not explicitly modeled. Hence, common causes of this type generally do not include a functional dependency of a component that is explicitly modeled even though loss of the supporting function (such as power or cooling) could cause multiple failures. The common

cause referred to here is an *implicit* dependency in the failures that can cause additional failures to be triggered once one failure occurs. CCFs must be considered and modeled in a fault tree because of their importance. Neglecting these CCF contributions can result in a significant underestimate of the probability of the top event. The key in successfully including CCFs in a fault tree is to identify the components that are susceptible to CCFs and then properly model them in the fault tree.

Before CCFs can be modeled properly they must be understood. Examples of CCFs that can occur and which have occurred are:

1. A common design or material deficiency that results in multiple components failing to perform a function or to withstand a design environment. Examples include undetected flaws in main engines and low material strengths in turbo pumps.
2. A common installation error that results in multiple components being misaligned or being functionally inoperable. Examples include check valves being installed backwards that remained undetected because they were not tested after installation.
3. A common maintenance error that results in multiple components being misaligned or being functionally inoperable. Examples include multiple valves remaining in a misaligned position after maintenance.
4. A common harsh environment such as vibration, radiation, moisture or contamination that causes multiple components to fail.

If a particular common cause such as a maintenance error is explicitly modeled in the fault tree then this particular common cause would not be modeled as a CCF. However, other CCF causes would still need to be included in the fault tree. Generally CCFs are potentially important contributors to the failure of redundant identical components, e.g., two identical motor operated valves of the same specification, two identical turbo pumps of the same specification, etc. Generally CCFs are more significant for active redundant components (motor operated valves, etc.) and become more dominant contributors as the number of redundant components increases.

The importance of CCF contributions can be illustrated by a simple example, which is representative of actual situations. Consider three redundant components that must all fail for the system to fail. The components can be relief valves, hydraulic pumps, controllers, or any other components providing a redundant function.

If each component has a failure probability p of 1 in a 1000, i.e. 1×10^{-3}, then the probability of the three components failing independently, $P_{independent}$ is

$$P_{independent} = p^3. \tag{5.1}$$

That is

$$P_{independentt} = 1\times10^{-3}\bullet1\times10^{-3}\bullet1\times10^{-3} \tag{5.2}$$

or

$$P_{independent}=1\times10^{-9}. \quad (5.3)$$

Consequently, the probability of the three components failing independently is 1 in one billion.

Now consider the possibility of a common cause failing all three components. Assume that there is a likelihood of 1%, i.e., 1×10^{-2}, that a common cause occurs. This can be interpreted as saying that in 1% of the failures there are common cause dependencies. This is the probability, for example, that a flaw exists in multiple components due to a manufacturing deficiency where the flaw has been missed by inspection. If the component fails from the deficiency then all the components with this deficiency will fail. The probability of the three components failing due to CCF, P_{CCF} is calculated as

$$P_{CCF} = 1\times10^{-3}\bullet1\times10^{-2} \quad (5.4)$$

or

$$P_{CCF} = 1\times10^{-5}. \quad (5.5)$$

Consequently P_{CCF} is one in one hundred thousand and is a factor of 10,000 greater than the independent failure probability $P_{independent}$. Note that in determining P_{CCF}, the individual component failure probability ($p=1\times10^{-3}$) is the first factor in Equation (4) and is multiplied by the probability of the CCF occurring which in this case is 1×10^{-2}. The common cause probability of 1×10^{-2} is a conditional probability of the remaining components failing given one component has failed. The common cause probability of 1×10^{-2} is equivalently the probability that the failure cause, given a failure, is a common cause that will affect all the other components. The CCF probability is thus a fraction of the failures involving all similar components. This is the basis for its estimation from data*.

As can be observed, because the CCF probability affects all components in a redundant set, even if it has a small likelihood of occurrence, it can enormously increase the probability of all the components failing and hence the system failing. Even if the CCF probability were 1 in 1000, i.e., 1×10^{-3}, the probability of all three components failing due to CCF would be 1×10^{-6} which is a factor of 1000 greater than the independent failure probability. This example is illustrative of the impacts of CCFs. Other specific CCF probabilities would produce other specific results, but the result would generally be larger than the independent probability of all the redundant components failing.

To include the CCF contribution in the fault tree, it must be separately modeled. For the three component redundancy example above, the fault tree structure could be as shown in Figure 5-3. alternately, the CCF contribution could be ORed with each individual component failure below the AND gate. This second approach is often more successful when modeling complex systems or when complex dependencies exist in a system.

* Several CCF models and associated data exist. The most common model is the β-factor model, in which β represents the fraction of the failure rate that is common to multiple susceptible components. More elaborate models have been developed that distinguish between CCF failures in different component size groupings, including the Multiple Greek Letter (MGL) model and the Alpha-factor model [1].

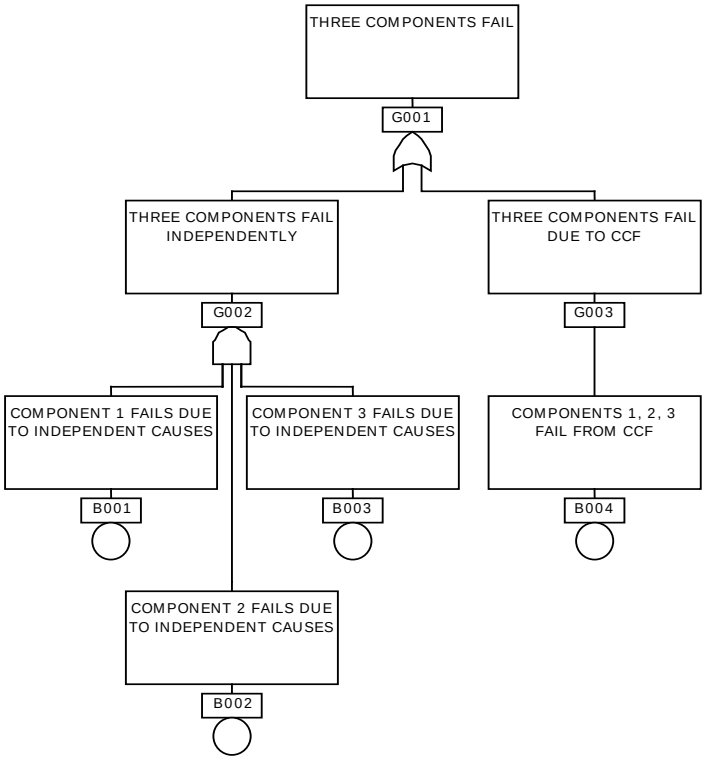


Figure 5-3. Fault Tree Representing Three Redundant Components with CCF Contribution.

In the actual fault tree, the identifiers of the components would be shown. What is important is that the CCF contribution is modeled as a separate contributor.

The same structure as above could be used for redundant trains of a system or redundant subsystems of a system. For example, instead of Component 1, 2, and 3, Train 1, 2, and 3 could be substituted. When the CCF at the train or subsystem level is modeled, it implies that the trains or subsystems are identical and that CCF data (e.g., the conditional CCF failure probability) is available at the train or subsystem level. If there is some doubt then CCF contributions should be modeled at the basic component level. A good rule of thumb is to include CCF contributions for any redundancy of identical, active components. If identical, redundant passive components are potentially important contributors then a CCF contribution can be included for these redundancies also. When in doubt, it is a good practice to model the CCF contribution in the fault tree. Sensitivity studies can be performed to determine the sensitivity of the top event probability to the CCF probability. If the top event probability is sensitive then more detailed analysis can be carried out.

5.3 Modeling Human Errors in a Fault Tree

Human performance plays a central role in overall safety. Human interactions with equipment occur during operations, response to accidents, surveillance testing, and maintenance. Human-machine interactions can mitigate the effects of accidents through recovery and control actions. Human interactions may also cause initiating events through errors. Humans are more perceptive

and flexible in performance than machines; hence, during adverse situations, crew members are expected to be able to deal with complex situations and system interactions. However, in the response errors may be committed that not only do not correct the situation but actually exacerbate it.

Human Reliability Analysis, or HRA, is a term applied to a series of methods used to describe qualitatively and quantitatively the occurrence of human errors that affect safety and reliability. Because human errors can be important contributors to risk, the inclusion of human error possibilities in FTA is important to provide a realistic picture of the overall failure probability and risk. Here, the focus is on what human errors should be modeled and how they should be modeled and not on how human errors are identified and quantified. Additional information on identification and quantification of human errors is included in [2] and [3].

Types of Human Errors

Examples of human errors that can be modeled include:

Test and maintenance related errors

Errors causing initiating events

Procedural errors during an incident or accident

Errors leading to inappropriate actions

Detection and Recovery errors

A brief description of each of these types of human errors to is presented so as to provide a better understanding as to why they may be of significance.

Test and Maintenance Related Errors

Test and maintenance related errors are errors induced by test and maintenance operations that can cause equipment to be degraded or to be put into an inoperable or disabled state. For example, these types of errors can occur in checkout and maintenance processes on the Space Shuttle. Examples of test and maintenance errors include:

Failing to properly align or restore a component or system following a test

Miscalibration of equipment

Incorrectly wiring a valve motor operator, contact or circuit

Incorrect re-assembly of a valve, pump, or component

Collateral damage resulting from maintenance performed on unrelated component, often in difficult work environments.

These human error contributions should be explicitly modeled in a fault tree if the equipment failure caused by the error is not modeled. If the equipment failure is modeled in the fault tree then the human error contribution need not be generally explicitly modeled. When the human error contribution is not explicitly modeled, then the failure probability assigned for the

equipment failure should include the inherent, equipment failure rate contribution plus the human error contribution. When the human error can result in a misconfiguration of the component, e.g., leaving a valve in a closed position, then the human error is generally explicitly modeled. This is because a misconfiguration is generally not the same as a failure and may have different recovery possibilities. Also a human error is always explicitly modeled when it can cause failure of two or more components in the tree. The human error is then explicitly modeled with the same identifier to identify its occurrence at multiple sites in the fault tree. If there is a question then the human error is modeled. The human error contribution is modeled in the fault tree as a basic event causing an equipment failure or equipment misconfiguration.

Errors causing initiating events

Errors causing initiating events include human actions that cause a fire to start, that cause a pipeline to break, and that cause a life support system to terminate. These types of human errors are generally included as causes of the initiating event frequencies used for quantification and hence are generally not explicitly modeled. A human initiating error is explicitly modeled in the fault tree if it is not included in the initiating event frequencies or if the human error can cause multiple impacts. Such multiple impacts occur, for example, if a human action can initiate an accident and at the same time impact a safety component that can help mitigate the accident.

Procedural errors during an incident or accident

Procedural errors can occur during an incident or accident in response to an off-normal situation. These are errors that may occur as an astronaut interprets the incoming diagnostic information, follows procedures, or takes action to implement an action. These errors are generally modeled in a fault tree assessing human procedural responses. The specific errors that are modeled are errors of omission in not responding, or not activating a system, or not changing the state of a component. The human error is modeled as a basic event in the fault tree with the equipment affected identified as the equipment failure and the cause attributed to human error.

Errors leading to inappropriate actions

Errors leading to inappropriate actions are sometimes termed errors of commission. These errors result in inappropriate actions being taken that may compound a problem. Errors of commission are sometimes committed in conjunction with procedural errors or errors of omission described in the previous paragraph. For example, instead of performing the steps in a procedure, different actions may be taken. Errors of commission are generally not modeled in a fault tree because of the difficulty of postulating a bounded scope for the possible errors of commission and the difficulty in assessing their probabilities.

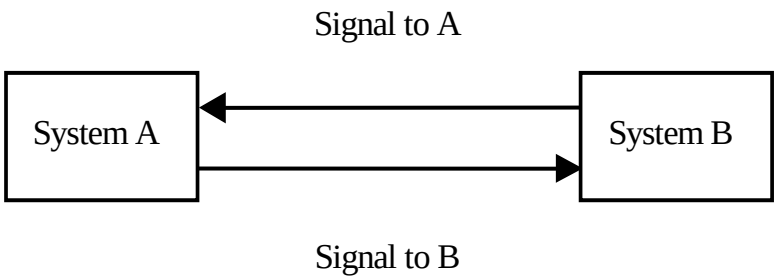
Detection and Recovery errors

Detection and Recovery errors are human errors to fail to detect and recover a failure that occurs. For example, a failed piece of equipment may be able to be repaired in time to help mitigate an accident if the appropriate human actions are taken. If these actions are not taken then an error occurs of failing to recover the failed piece of equipment. A recovery error also occurs if a human error in following a procedure is not caught. Detection errors can occur if defects or failures are not detected in inspections and tests. Detection and recovery errors are sometimes

modeled in the fault tree to show their particular contributions. When these errors are not explicitly modeled then the probabilities of failing to detect and restore failed equipment are included in the equipment failure probability quantification. In defining the scope and approaches for the fault tree, as part of the steps of the fault tree process, it is important to define how such human errors, and all the types of human errors, will be modeled.

5.4 Modeling Loops and Feedback

Consider the feedback loop illustrated below. In this figure, System A sends a signal to System B.



System B in turn provides a signal to System A. For example, System A could be a controller sending a control signal to System B, and System B an online system sending a feedback signal to System A. As a more specific example, for the Space Shuttle, System A could be the Orbiter, which sends a control signal, and System B could be the Main Engine, which provides the feedback signal.

For a fault tree model of a feedback situation such as this, it is desirable that the model include system failures that include the failure of a signal from another system. However, it is also necessary to avoid feedback loops in the fault tree model. These loops will occur if the failure of System B is included as a contributor to System A failing *and* the failure of the signal from System A is included as a contributor to System B. In this case the model would indicate System A causing the failure of System A. A similar loop would occur for System B.

A fault tree model for System B failing and a fault tree model for System A failing that avoid these loops are shown in Figure 5-4.

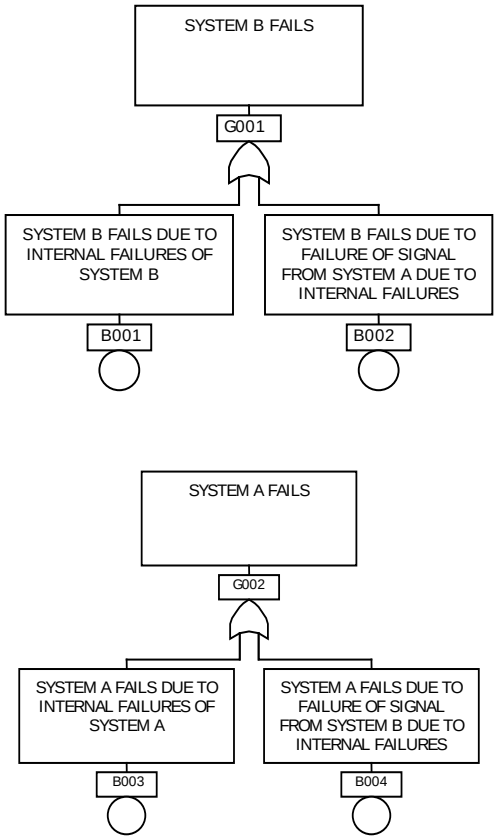


Figure 5-4. Fault Tree Models that Avoid Logical Loops

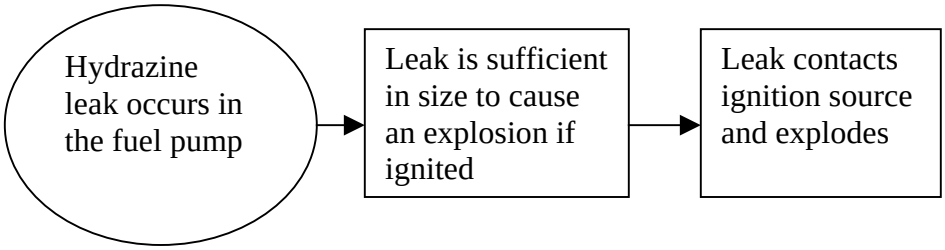
In the above fault tree models only internal failure causes are included and not failures due to any feedback. For example, causes for System A failing from a signal failure of System A should not be included. Thus the loops are cut (eliminated) in the fault tree. In the above fault tree for System A failure, for example, for the fault on the right hand side of the OR gate, the tree will only be developed to include internal failures of B. A failed signal from A will not be included as a cause of B failing. This also applies to the model for System B failing. It should be noted that the internal failures for a given system might be different for the two trees because of the different failures considered. However, again, only internal failures of the system originating the signal are considered. This general rule of considering internal failures of the system originating the signal applies to any number of systems interacting with each other. If this general rule is followed, then no loops will be constructed in the resulting fault tree.

5.5 Modeling of a Phenomenological Sequence

A phenomenological sequence consists of an initiating event and additional events, called pivotal events, that lead to an undesired event. A phenomenological sequence is more often modeled using an Event Sequence Diagrams (ESD) or an Event Tree (see Chapter 9). However, there are instances where modeling the sequence using a fault tree might be desired. In this case, the final undesired event in the sequence is the same as the top event of the fault tree. The sequence can

be modeled in the fault tree using standard fault tree modeling techniques. However care must be taken in describing the events in the fault tree to show their conditional dependency.

To illustrate the considerations in fault tree modeling of a sequence, consider the figure below that illustrates a sequence for a hydrazine leak resulting in an explosion.



The first event is the occurrence of a hydrazine leak aboard the craft. The second event is the condition that the leak is a sufficiently large leak to have the potential for an explosion. The third event is the event of the hydrazine leak contacting an ignition source given the leak is of sufficient size.

The conditional nature of the second and third events are implied by the sequence diagram. In modeling the events in a fault tree, this conditional nature must be explicitly defined to identify the order of the events when the fault tree is evaluated. By defining these given conditions, the probabilities can then be calculated and the events arranged in proper order. Figure 5-5 illustrates the corresponding fault tree model for this sequence.

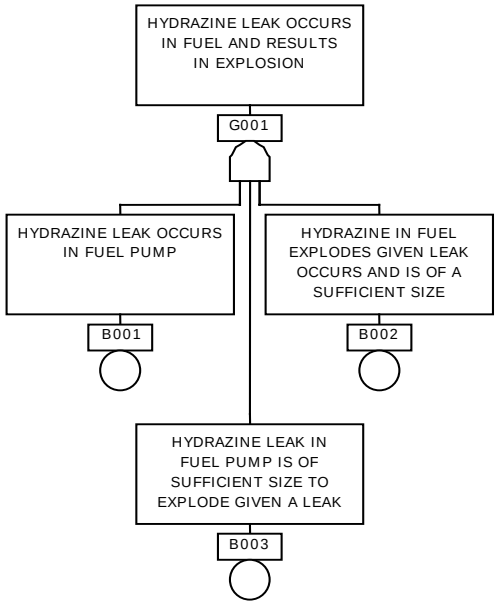


Figure 5-5. Fault Tree for Explosion Resulting from Hydrazine Leak

As observed in the above fault tree structure, the sequence is modeled as an AND gate. The descriptions of the basic events can vary, however the conditional nature of the events need to be clearly identified in the event descriptions. As shown above, the events are clearly defined, identifying the substance leaking and the component affected, and the given conditions leading up to the event.

The given conditions of the above events define the order of the events. The probabilities assigned to the events will be conditional probabilities, conditioned on the previous events having occurred as defined by the given conditions. If a more explicit model is desired then the AND gate needs to be replaced by a PRIORITY AND gate which will assure that the given physical order of the events is retained.

5.6 Naming Schemes for the Fault Tree

The critical need in naming gates and basic events is to ensure that correct cut sets and probabilities are calculated if a gate or basic event occurs more than once in the model. If a basic event representing an identical component failure has different names a cut set calculation for an AND gate would erroneously conclude that a single failure is a double failure. Similarly, if the events were to occur under an OR gate the event would be double counted. Basic event and gate names are used to input the fault tree to a computer code. These names are identifiers of each event in the tree. Any identifiers can be used, but they should be concise and descriptive. FT software generally limits the number of characters that can be used. Common limits are 18 or 24 characters. For a component failure, the identifier often used is of the form: Component Type-Component ID-Failure Mode. The Component Type is the general type of component. Examples are MOV for motor operated valve, PMP for pump, and TNK for tank, The Component ID is the identifier of the unique component that is often related to a system or schematic number. Example failure modes are FTO for fail to open, FTC for fail to close, and INOP for inoperable. Thus, a basic event could be identified as MOV-1233-FTO. If different systems were being modeled then a system identifier could be the first identifier. An alternate naming scheme used in the Space Shuttle PRA places the system identifier differently: ElementID-SystemID-ComponentID-FailureMode. Table 5-1 shows other representative identifiers that have been used.

Table 5-1. Representative Component and Failure Mode Identifiers

Component Type	Component Failure Mode	Description
HX	F	Heat Exchanger Cooling Capability Fails
HX	J	Heat Exchanger Tube Rupture
HX	P	Heat Exchanger Plugs
IN	F	Inverter No Output
IR	F	Regulating Rectifier No Output
IV	F	Static Voltage Regulator No Output
LC	D	Logic Circuit Fails to Generate Signal
LS	D	Level Switch Fails to Respond
LS	H	Level Switch Fails High
LS	L	Level Switch Fails Low

Intermediate events in the fault tree can have similar naming conventions, with each identifier identifying the type of event and the input events. In some past FTAs, gate names were not codified to the same degree as the names for the basic failures and basic events. This is because the basic failures and basic events are the principal outputs of fault tree computer codes. What is most important is that an agreed upon naming scheme be used for the events before the fault tree is constructed, and that procedures exist to minimize the likelihood of inadvertently giving the same basic event two names. Going through an already constructed fault tree and changing names is a challenging and difficult task, as anyone who has attempted it will attest, and it will be likely to result in errors.

5.7 Fault Tree Construction Ground Rules

In constructing a fault tree, ground rules are oftentimes used to narrow the scope or resolution of the fault tree without impacting the overall results. Ground rules listed below have been used in past fault tree analyses. These ground rules should be evaluated for their applicability to the particular problem being addressed before they are applied.

1. Model to the highest level for which data exists and for which there are no common hardware interfaces with other contributors. In applications, this generally means modeling to the major component level, e.g. a motor operated valve, high-pressure pump, etc. Modeling to a lower level will not only be a waste of time but will often provide erroneous probabilities or probabilities with much larger uncertainties. This is an example of the fault tree maxim- “too much detail, too much uncertainty”.
2. Do not model wiring faults between components. Generally, wiring faults, such as shorts to ground and shorts to power, have very low probabilities compared to probabilities of major components failing. However, if there are no significantly

higher contributors or if the wiring can be impacted by other failures (e.g., a fire) then wiring faults need to be considered. Also if the objective includes the modeling of wiring faults then they need to be included (for example, in the case of wear over time of wiring bundles due to maintenance activity on a reusable vehicle).

3. Do not model piping faults between components. The reasoning here is similar to Item 2. Piping faults generally have very small probabilities of occurring. However, if there are no significantly higher contributors or the objective includes modeling piping faults (e.g., then they need to be included because aging effects increase the global impact of piping failure over time).
4. Do not model out-of-design conditions. In general, the component is not intended to operate outside of design conditions. Therefore, these conditions are not generally modeled and out-of-design failures are not shown on the fault tree. In special circumstances these may be included but in these circumstances justification must be provided.
5. Model CCF contributors on all identical active, redundant components. Active components are generally mechanical or electro-mechanical components that change state. Examples are pumps, motor operated valves, motors, and circuit breakers. Generally CCF contributors are highest for these active components because of their interactions with environments and operations. CCF contributors can be included for identical, redundant electrical components or passive components if these components are assessed to be possibly susceptible to common failure causes. In past FTAs, CCF contributors have often been included on all identical, redundant components and sensitivity analyses performed to identify those CCF contributors that could most influence the results. These sensitive CCF contributors were then analyzed in more detail.
6. Do not model human errors of commission. Human errors of commission are those involving the human committing an unforeseen action. The reason human errors of commission are not modeled is that current modeling approaches would require a consideration of an almost unlimited scope of actions.
7. Do not continue to model an AND gate with four or more inputs if there are triple, double, or single contributors elsewhere in the tree *and* if it can be assured that there are no common hardware interfaces to the inputs. Sometimes instead of four or more inputs, AND gates are not continued if they have three inputs. This ground rule is a common sense type rule that says don't chase higher order combinations of faults if there are lower order combinations already identified. This rule has to be evaluated for its applicability for each situation. For these cases, the events inputting the AND gate are kept as undeveloped events that could be expanded later if necessary.
8. Do not continue to model an input to an OR gate if there is information that assures its probability is significantly lower than the probability of one or more of the other inputs. This rule is a common sense rule like the previous rule. This rule suggests not chasing contributors that can be shown to be significantly lower in probability than the contributors already identified. The input events not continued are again shown as undeveloped events.

5.8 Validating a Fault Tree

The topic of validating a fault tree is an important one but a difficult one. Therefore it is not often addressed in textbooks or papers. Validating a fault tree is difficult since the quantitative result of the tree is a probability, which is not at all tangible. However, the fault tree can be validated at least indirectly using the following steps:

1. Obtain the minimal cut sets of the fault tree. Identify the smallest order cut sets and check if these are indeed valid failure paths to the top event. Use the system schematic or system diagram for these checks. An FMEA may also be used for these checks if there are only one or two components in the failure paths that are checked.
2. Identify failures and basic events that have occurred as recorded in databases. Check if these have been included in the fault tree and if not, why not. This is a check on the scope and completeness of the fault tree.
3. Obtain the success paths of the fault tree. Identify the smallest of these success paths and validate that these are indeed success paths. This is similar to Step 1 but uses the success paths to cross-check the fault tree.
4. Identify lower order faults (i.e., intermediate events) in the fault tree. Obtain the cut sets and success paths for these lower order faults. Validate these cut sets and success paths as in Steps 1 and 2. Validating lower order faults is done if validating the top event directly in Steps 1 and 2 is too difficult, e.g. the smallest order cut sets contain a large number of basic events that are difficult to check.
5. Check the probabilities of the cut sets and their relative contributions to determine if the results are reasonable. This is a sanity check on the reasonableness of the cut sets probabilities. The lowest order cut sets and those containing active component failures should generally have the highest probabilities. Since each cut set is similar to a parallel system of the contributors, the probability value can be compared with values from past experience. If CCF contributors or human errors have been modeled then these should appear as relatively high contributors.
6. Check the probabilities of intermediate faults for their reasonableness. These intermediate faults are often subsystem or module faults. Their values can be compared with those from past experience. Since intermediate faults will generally have higher probabilities than the probability for the top event, these intermediate fault probabilities can often be compared with failure data involving the intermediate faults.
7. Check the probability of the top event for its reasonableness. Compare it to results obtained for similar type assessments that have been performed in the past. Very low probabilities in particular, such as 1×10^{-9} or lower, are fictitious and generally only show that the more likely ways of the top event occurring have not been identified in the fault tree. (Remember, the chance of a meteoroid or other catastrophe destroying the earth is of the order of 1×10^{-9}.)

5.9 References

1. A. Mosleh, *Procedure for Analysis of Common-Cause Failures in Probabilistic Safety Analysis*, NUREG/CR-5801, US Nuclear Regulatory Commission, 1993.
2. D. Gertman and H. Blackman, Human Reliability and Safety Analysis Data Handbook, John Wiley and Sons, 1994.
3. E. Dougherty and J. Fragola, *Human Reliability Analysis: A Systems Approach with Nuclear Power Plant Applications*, Wiley, New York, 1988.

6. Qualitative Evaluations of a Fault Tree and Basic Probability Formulas

The following sections describe qualitative evaluations of a fault tree. These evaluations include applying Boolean algebra to the fault tree to obtain the equations for each gate of the fault tree. The associated basic probability formulas for each type of gate are described. The section also describes a methodology for obtaining the minimal cut sets of the fault tree and minimal path sets from a complimentary success tree, as well as an alternate approach to solving a fault tree using binary decision diagrams.

6.1 Application of Boolean Algebra in Fault Tree Analysis

A fault tree, as is now understood, is a logic diagram depicting certain events that must occur in order for other events to occur. The events are termed "faults" if they are initiated by other events and are termed "failures" if they are the basic initiating events. The fault tree interrelates events (faults to faults or faults to failures) and certain symbols are used to depict the various relationships. As described in Chapter 4, the basic symbol is the "gate" and each gate has inputs and an output as shown in Figure 6-1.

The gate output is the "higher" fault event under consideration and the gate inputs are the more basic ("lower") fault (or failure) events that relate to the output. When a fault tree is drawn, the tree is developed from the "higher" faults to the more basic faults (i.e., from output to inputs). In this process, certain techniques are used to determine which category of gate is appropriate. The two basic gate categories are the OR-gate and the AND-gate. Because these gates relate events in exactly the same way as Boolean operations, there is a one-to-one correspondence between the Boolean algebraic representation and the fault tree representation. The rules of Boolean Algebra are summarized in Appendix A.

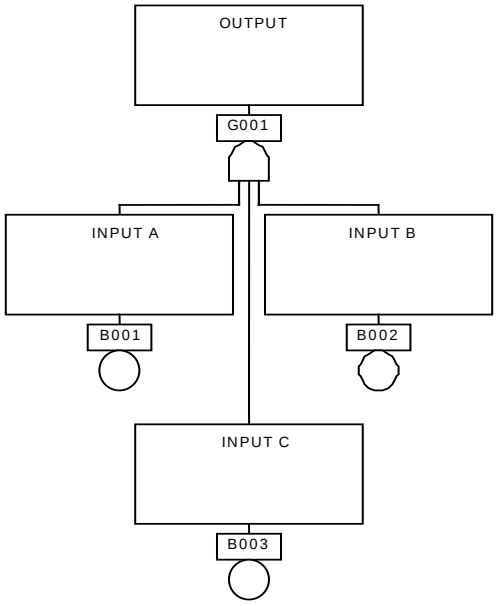
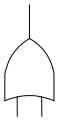


Figure 6-1. The Gate Function in a Fault Tree

The OR-Gate



As described in Chapter 4, the fault tree symbol is an OR-gate which represents the union of the events attached to the gate. Any one or more of the input events must occur to cause the event above the gate to occur. The OR-gate is equivalent to the Boolean symbol "+." For example, the OR-gate with two input events, as shown in Figure 6-2, is equivalent to the Boolean expression, Q=A+B. Either of the events A or B, or both must occur in order for Q to occur. Because of its equivalence to the Boolean union operation denoted by the symbol "+," the OR-gate is sometimes drawn with a "+," inside the gate symbol as in Figure 6-2. For n input events attached to the OR-gate, the equivalent Boolean expression is $Q=A_1+A_2+A_3+\ldots+A_n$.

In the terms of probability,

$$P(Q) = P(A) + P(B) - P(A \cap B) \qquad (6.1)$$

or

$$= P(A) + P(B) - P(A)P(B|A)$$

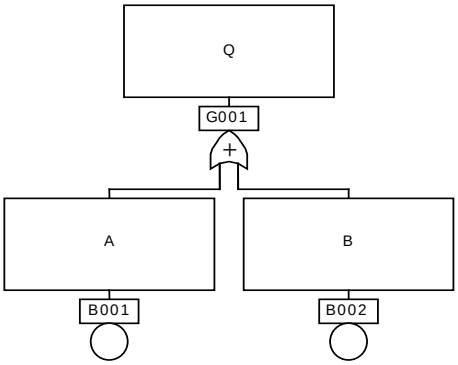


Figure 6-2. A Two-Input OR-Gate

The following observations can be made:

- If A and B are mutually exclusive events, then $P(A \cap B) = 0$ and $P(Q) = P(A) + P(B)$;
- If A and B are independent events, then $P(B|A) = P(B)$ and $P(Q) = P(A) + P(B) - P(A)\,P(B)$;
- If event B is completely dependent on event A, that is, whenever A occurs, B also occurs, then $P(B|A) = 1$ and

$$P(Q) = P(A) + P(B) - P(A)$$

$$P(Q) = P(B);$$

- The approximation $P(Q) \cong P(A) + P(B)$ is, in all cases, a conservative estimate for the probability of the output event Q, i.e.,

$$P(A) + P(B) \geq P(A) + P(B) - P(A \cap B) \text{ for all A, B;}$$

- If A and B are independent, low probability events (say $P(A), P(B) < 10^{-1}$), then $P(A \cap B)$ is small compared with $P(A) + P(B)$ so that $P(A) + P(B)$ is an accurate approximation of P(Q).

- In an EXCLUSIVE OR-gate with two inputs A and B, the output event Q occurs if event A occurs or event B occurs, but not both. The probability expression for the output event Q of an EXCLUSIVE OR-gate is:

$$P(Q)_{\text{EXCLUSIVE OR}} = P(A) + P(B) - 2P(A \cap B). \quad (6.2)$$

Comparing Equations (1) and (2), we observe that if A and B are independent low probability component failures, the difference in probability between the two expressions is negligible. This is why the distinction between the inclusive and exclusive OR-gates is generally not necessary in FTA where independent, low probability component failures are often under consideration. It may sometimes, however, be useful to make the distinction in the special case where the exclusive OR logic is truly required, and in addition where there is a strong dependency between the input events or the failure probabilities are high. In this latter case, the intersection term may be large enough to significantly effect the result (if this is the case the events can be made mutually exclusive by replacing each event by an AND gate with one event the compliment of the other event as the inputs). In conclusion, it should be observed that in any case, the error which is made by using the inclusive rather than the exclusive OR-gate biases the answer on the conservative side because the inclusive OR has the higher probability. In the remainder of this text, unless otherwise noted, all references to the OR-gate should be interpreted as the inclusive variety.

Figure 6-3 shows a realistic example of an OR-gate for a fault condition of a set of normally closed contacts.

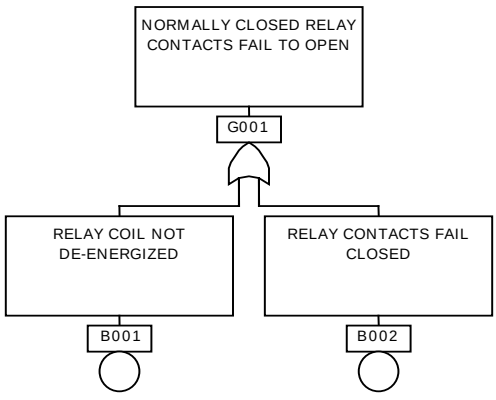


Figure 6-3. A Specific Two-Input OR-Gate

This OR-gate is equivalent to the Boolean expression

RELAY CONTACTS FAIL TO OPEN	=	RELAY COIL NOT DE-ENERGIZED	+	CONTACTS FAIL CLOSED

Instead of explicitly describing the events, a unique symbol (Q, A_2, etc.) is usually associated with each event as shown in Figure 6-2. Therefore, if the event "relay contacts fail to open" is labeled "Q", relay coil not "de-energized" is labeled "A," and "contacts fail closed" is labeled "B," the OR-gate of Figure 6-3 can be represented by the Boolean equation Q=A+B.

An OR-gate is merely a re-expression of the event above the gate in terms of the more elementary input events. The event above the gate encompasses all of these more elementary events; if any one or more of these elementary events occurs, then B occurs. This interpretation is quite important in that it characterizes an OR-gate and differentiates it from an AND-gate. The input events to an OR-gate do not cause the event above the gate; they are simply re-expressions of the event above the gate. This topic has been addressed previously, but it is considered so important to fault tree analysis that it is covered again, having now reviewed the concepts of Boolean algebra.

Consider two switches in series as shown in Figure 6-4. The points A and B are points on the wire. If wire failures are ignored then the fault tree representation of the event, "No Current to Point B" is shown in Figure 6-5.

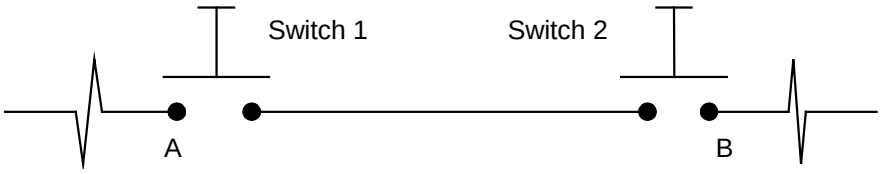


Figure 6-4. Two Switches in Series

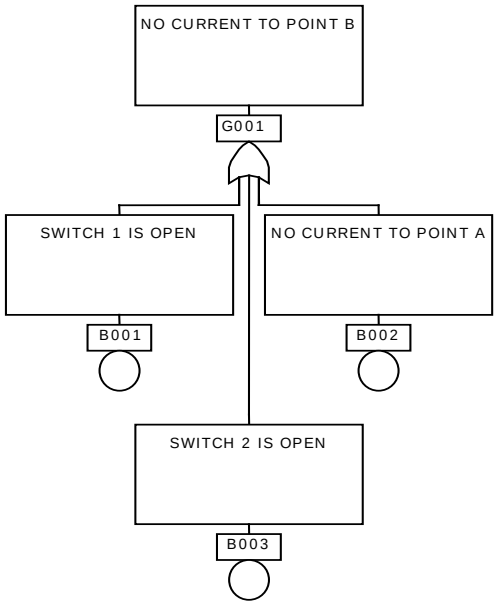


Figure 6-5. A Specific Three-input OR-Gate

If the events are denoted by the symbols given below, then the Boolean representation is

$$B = A_1 + A_2 + A_3.$$

NO CURRENT TO POINT B = B

SWITCH 1 IS OPEN = A_1 SWITCH 2 IS OPEN = A_2 NO CURRENT TO POINT A = A_3

The event B occurs if A_1 or A_2 or A_3 occurs. Event B is merely a re-expression of events A_1, A_2, A_3. We have classified the particular events A_1, A_2, A_3 as belonging to the general event B.

The AND-Gate

As discussed in Chapter 4, the fault tree symbol  is an AND-gate which represents the intersection of the events attached to the gate. The AND-gate is equivalent to the Boolean symbol "•." All of the input events attached to the AND-gate must exist in order for the event above the gate to occur. For two events attached to an AND-gate (Figure 6-6), the equivalent Boolean expression is Q=A•B. Because of its equivalence to the Boolean intersection operation denoted by the symbol "•", that symbol is sometimes included inside the AND-gate as in Figure 6-6. For n input events to an AND-gate, the equivalent Boolean expression is

$$Q = A_1 \bullet A_2 \bullet A_3 \bullet \ldots \bullet A_n.$$

In this case, event Q will occur if and only if all the A_i occur. In terms of probability, for two events

$$P(Q) = P(A)P(B|A) = P(B)P(A|B). \quad (6.3)$$

The following observations can be made:

- If A and B are independent events, then P(B|A)=P(B), P(A|B)=P(A),

  and P(Q)=P(A) P(B);

- If A and B are not independent events, then P(Q) may be significantly greater than P(A)P(B). For example, in the extreme case where B depends completely on A, that is, whenever A occurs, B also occurs, then P(B|A)= 1 and P(Q) = P(A).

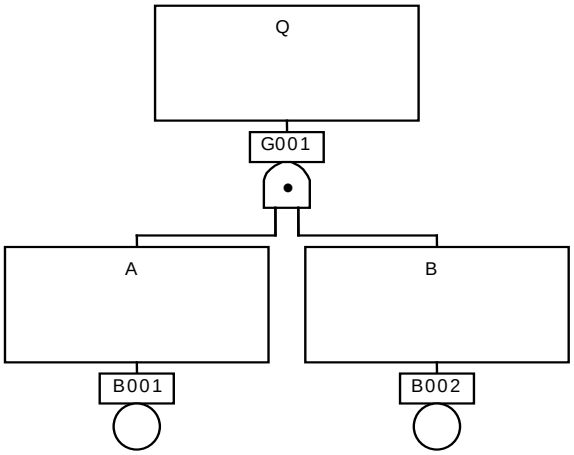


Figure 6-6. A Two-Input AND Gate

The events attached to the AND-gate are the causes of the event above the gate. Event Q is caused only if every one of the input events occurs. This causal relationship is what differentiates an AND-gate from an OR-gate. If the event above the gate occurs when any one of the input events occurs, then the gate is an OR-gate and the event is merely a restatement of the input events. If the event above the gate occurs only when combinations of more elementary events occur, then the gate is an AND-gate and the inputs constitute the cause of the event above the gate.

This discussion is concludes with an example indicating how Boolean algebra can be used to restructure a fault tree. Consider the equation D = A•(B+C). The corresponding fault tree structure is shown in Figure 6-7.

Now according to Rule 3a in Appendix A, event D can also be expressed as D = (A•B) + (A•C). The fault tree structure for this equivalent expression for D is shown in Figure 6-8.

The two fault tree structures in Figures 6-7 and 6-8 may appear to be different; however, they are equivalent. Thus, there is not one "correct" fault tree for a problem but many correct forms that are equivalent to one another. The rules of Boolean algebra can thus be applied to restructure the tree to a simpler, equivalent form for understanding or for simplifying the evaluation of the tree. Later, the rules of Boolean algebra will be applied to obtain one form of the fault tree, called the minimal cut set form, which allows quantitative and qualitative evaluations to be performed in a straightforward manner.

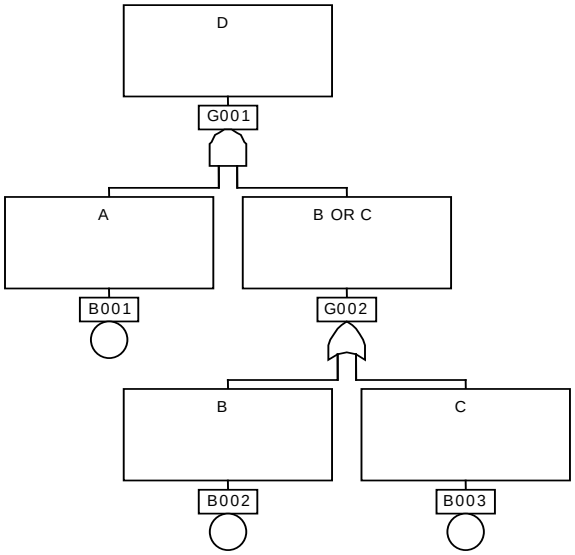


Figure 6-7. Fault Tree Structure for D = A•(B+C)

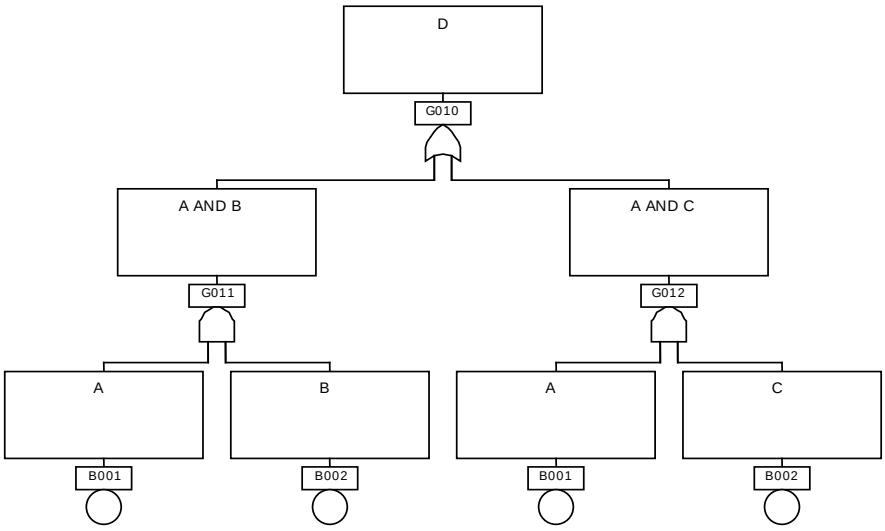


Figure 6-8. Equivalent Form for the Fault Tree of Figure 6-7

6.2 Binary Decision Diagrams

Recent developments in digital logic have helped produce an alternative analysis procedure for fault trees. This alternative approach, based on the use of Binary Decision Diagrams (BDDs), works directly with the logical expressions instead of the cut sets. A BDD can be thought of as a graphical representation of a data structure for a logic function.

The BDD that is used for fault tree analysis is more properly referred to as a Reduced Ordered BDD. Reduced means that the BDD is in minimal form. Ordered means that the variables appear in the same order on each path. For more information on the BDD approach to fault tree analysis see [1], [2].

The BDD is constructed from the fault tree recursively, which can be seen as being bottom-up. Each basic event has an associated single-node BDD. For example, the BDD for a basic event B is shown in Figure 6-9. Starting at the bottom of the tree, a BDD is constructed for each basic event, and then combined according to the logic defined by the gate. The BDD for the OR relation B + C is constructed by applying the OR function to the BDD for B and the BDD for C. Since B is first in the relation, it becomes the "root" node. The C BDD is then OR'ed with each "child" node of B. Thus, as shown in Figure 6-10, B is the root node and the C BDD is ORed with the left and right children of B.

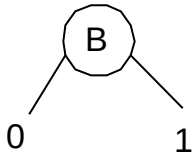


Figure 6-9. BDD for Basic Event B

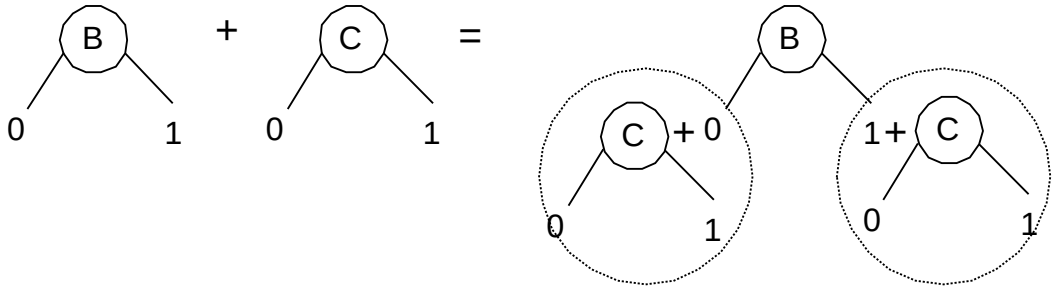


Figure 6-10. OR of BDD for B and C (step 1)

First consider the left child of B, which is terminal node 0. From the laws of Boolean algebra (8b and 8d in Appendix A, Table A-1), 0+X = X and 1+X = 1. Thus, the left child reduces to C and the right child reduces to 1 (See Figure 6-11).

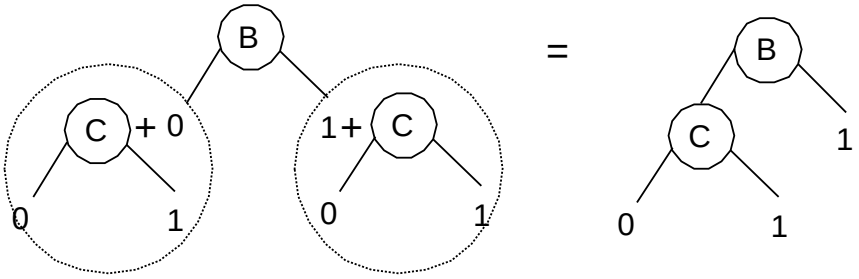


Figure 6-11. OR of BDD for B and C (step 2)

Next consider the AND operation applied to events A and B, shown in Figure 6-12. Again from the laws of Boolean algebra, 0•X = 0 and that 1•X = X. Thus the reduced BDD for event A•B is shown in Figure 6-13.

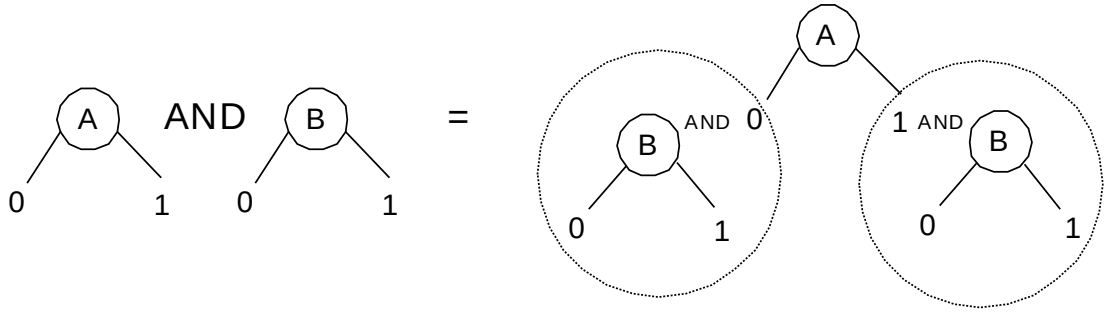


Figure 6-12. AND of BDD for B and C (step 1)

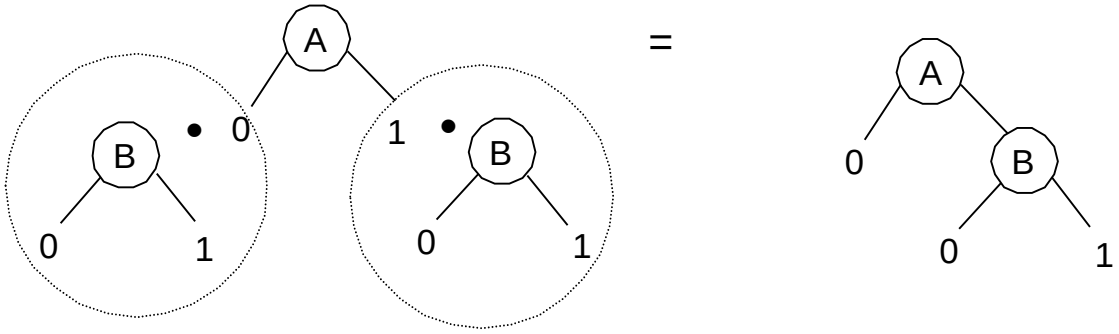


Figure 6-13. AND of BDD for A and B (step 2)

Consider now a basic event C that is ORed with the AND gate of A and B, i.e. A•B+C. The BDD construction is shown in Figure 6-14. Since A comes before C, A becomes the root node and the OR operation is applied to A's children. The left child reduces by Boolean algebra and the right child continues as in Figure 6-11 producing the BDD shown in Figure 6-15.

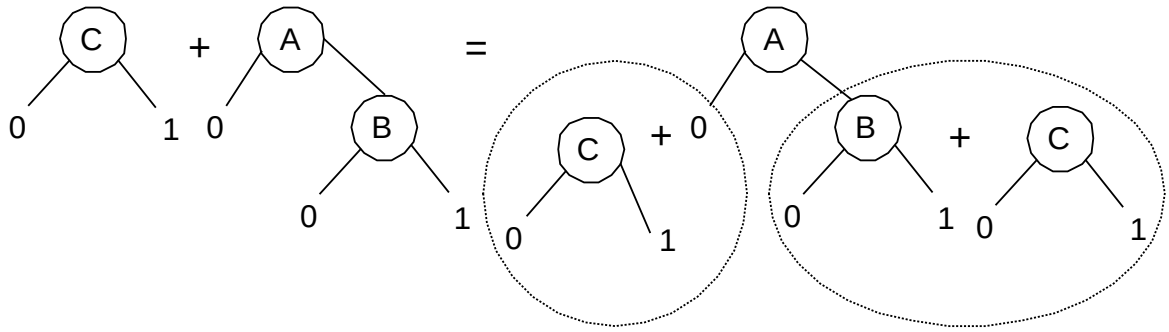


Figure 6-14. BDD construction for an OR with an AND gate (step 1)

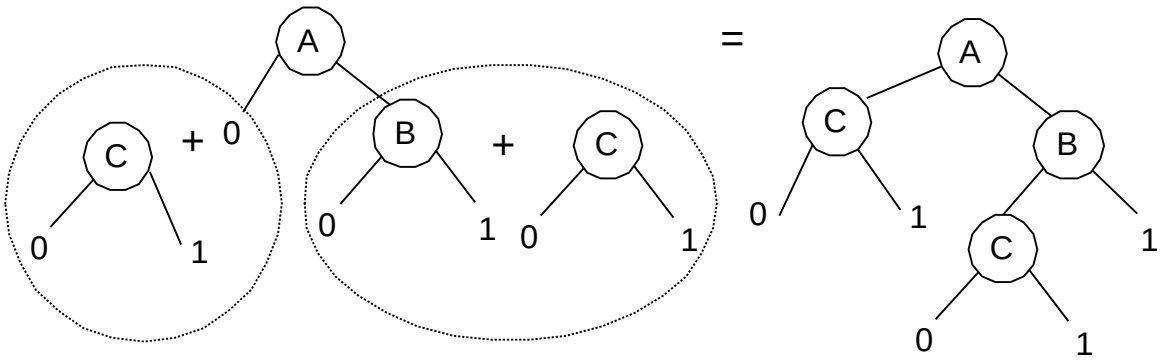


Figure 6-15. BDD construction for an OR with an AND gate (step 2)

The BDD on the right side can be reduced further. Notice that there are two identical instances of the node representing C. One is redundant and can be removed, as shown in Figure 6-16.

Each path from the root node to a terminal node with value 1 represents a disjoint combination of events that causes system failure. Thus, if Figure 6-14 represented a small fault tree of C ORed with A AND B the failure paths would be A'C+AB+AB'C. Since the paths are disjoint, the calculation of the probability of failure is straightforward—the probability of failure is the sum of the probabilities associated with the paths.

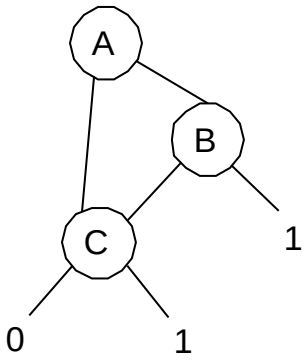


Figure 6-16. BDD construction for an OR with an AND gate (step 3)

The transformation of a fault tree would proceed in the above manner until all the gates have been linked in the BDD. The failure paths leading to an end value of 1 would be quantified by summing the failure paths that by their nature are disjoint. In developing the BDD for a fault tree various reduction techniques are used to simplify the BDD. For example, the above BDD can be further reduced because of the two identical nodes representing C.

6.3 Comparison of the BDD Approach with the Minimal Cut Set Approach

The BBD approach is a complementary approach to the MCS approach. Each approach has its advantages and features. The MCS approach identifies the minimal sets (combinations) of basic events that will cause the top event. The MCSs thus highlight the most significant failure combinations and show where design changes can eliminate or reduce undesirable combinations. MCSs also support fault tree validation in that specific minimal cut sets can be checked to determine if they indeed cause the top event. MCSs support recovery actions—effective recovery procedures are aimed at recovering at least one failure in the dominant minimal cut sets. MCSs can furthermore be reviewed for dependencies and susceptibilities to CCF potentials. The MCSs thus provide valuable, qualitative information as well as quantitative information.

The BDD approach provides an exact calculation of the top event probability. The exact probability is useful when many high-probability events appear in the model. The BDD approach is also the most efficient approach for calculating probabilities. Because the minimal paths generated in the BDD approach are disjoint, calculation of importances and sensitivities can be done in an efficient and exact manner.

For very large fault trees having many AND and OR gates, in which many MCSs can be generated, the MCS approach must often truncate the lowest probability minimal cut sets to calculate the probability of the top event in relatively short time. The result of this calculation is generally accurate to at least two significant figures, which is typically more accurate than the basic event probabilities that are used once their uncertainties are considered. Present fault tree software packages have algorithms for bounding the truncation error. If this error is too large then the truncation limit can be lowered and more minimal cut sets generated. Because of the speed of present personal computers, generating sufficient numbers of MCSs or generating more MCSs if needed is usually not a problem.

The BDD approach is thus more efficient and precise in quantifying probabilities and importances. The MCS approach provides important qualitative information as well as quantitative information. The most information is provided by using both approaches. The use of BDDs does not preclude the determination of MCSs. Many available software packages only use the MCS approach, which has been the standard fault tree evaluation approach for many years. However, there are software packages available that use the BDD approach, and a few use both approaches. In the future, more software packages are expected to include both approaches for evaluation.

6.4 References

1. R. Sinnamon and J. Andreas, *Fault Tree Analysis and Binary Decision Diagrams*, Proceedings of the Reliability and Maintainability Symposium, January 1996, pp 215-222.

2. A. Rauzy, *New Algorithms for Fault Tree Analysis*, Reliability Engineering and System Safety, Vol. 40, 1993, pp 203-211.

7. Quantitative Evaluations of a Fault Tree

7.1 Basic Quantification of the Fault Tree and Associated Data Used

To quantify the probability of the top event of the FT a probability for each basic event (BE) in the fault tree must be provided. These BE probabilities are then propagated upward to the top event using the Boolean relationships for the FT. This process was illustrated when the basic gates of the FT were discussed in Chapter 6. The BE probabilities can be propagated upward using BDDS that represent the FT structure. Alternatively, the minimal cut sets can be generated from the FT and then used to quantify the top event. The minimal cut set generation approach is used by most FT software because of the additional, important information provided by the minimal cut sets.

Since the top event is expressed as the union of the minimal cut sets, the probability of the top event can be approximated as the sum of the individual minimal cut set probabilities, provided these probabilities are small. This is typically an accurate approximation for basic event probabilities below 0.1. The approximation is termed the "rare event approximation." Other, more accurate approximations can be applied if needed using standard bracketing approaches to compute the probability of a union. However, FT software often uses the simple sum of the minimal cut set probabilities as the standard quantification method. A user must be aware of this, and use a more accurate quantification method, considering the intersection of the minimal cut sets, if minimal cut set probabilities are high (if the sum of the minimal cut set probabilities is greater than 0.1).

Since a minimal cut set is an intersection of BEs, the probability of a minimal cut set is simply the product of the individual BE probabilities. Thus, the probability of the top event is expressible as the sum of the products of individual BE probabilities. This expression is called the sum of products approximation. It has a relative accuracy of at least 10% (at least two significant figures) if the BE probabilities are less than 0.1. Further, even if some BE probabilities are greater than 0.1, the approximation is accurate if most of the probabilities are less than 0.1. The accuracy can be checked by carrying out a second order bracketing approximation (i.e., including the intersections between all pairs of minimal cut sets) and comparing the results for the top event. Additionally some FT codes estimate the accuracy of the sum of products approximation.

In terms of symbols the sum of products expression is given as:

$$P(\text{Top}) = \Sigma\, P(M_i) \tag{7.1}$$

$$P(M_i) = P(BE_1)P(BE_2) \ldots P(BE_k) \tag{7.2}$$

where the term "P( )" denotes the probability of the enclosed event, "Top" denotes the top event, "M_i" denotes a particular minimal cut set, "BE" a basic event, and k the number of basic events in a minimal cut set. The symbol "Σ" is a summation sign and denotes the summation over the number of minimal cut sets that are evaluated.

If the FT is a small one and has relatively few events then all the minimal cut sets of the FT can be generated. If the FT is large, say more than 100 events, and has AND gates with OR gate inputs then the number of minimal cut sets for the FT can be too large to be exhaustively generated. For some large FTs that have been constructed, the numbers of minimal cut sets have exceeded one million. Most FT software includes techniques for estimating the total number of minimal cut sets in the FT based on the number of BEs, the number of gates and the types of gates. When the number of minimal cut sets is large then the number generated can be truncated for the sum in Equation (1). In this case, using the probabilities of the BEs and the logic structure of the FT, minimal cut sets are not generated if their probability is below some cutoff value, such as 1×10^{-12}. The minimal cut set probability can be estimated (bounded) using the logic structure of the FT and probability bounding techniques. FT software documentation describes these techniques.

The input data that must be supplied for a BE is usually one of four basic types:

1) a *component failure probability* in some time interval,
2) an *event occurrence probability* in some time interval,
3) a *component unavailability*, and
4) a *pure event probability*.

To calculate the *component failure probability* over some interval, a *component failure rate (expressed in failures per unit time)* and the *elapsed mission time* must be supplied as input data. The elapsed mission time is the sum of the entire time over which the component is in an operating and non-operating state. The operating state is defined as a state where the component is subjected to the stresses of actual operation that may involve the transfer of energy, signals or information. The non-operating state is defined as the state when the component has been "shutdown" and is subject only to environmental stresses due to its location within a system. The failure rate can be defined as

$$\lambda = \lambda_O d + \lambda_N(1-d),$$

where

d = fractional duty cycle (total operating time/total mission time)

λ_O = the component failure rate in the operating state

λ_N = the contribution to the component failure rate from the non-operating state.

The standard assumption is to assume a constant failure rate. This is based on the further assumption that there is no aging (wearout) and that burn in of the component was 100%

effective and removed all infant mortalities from the population of parts used for assembly, and therefore the failures are purely random*.

The foundation of a good analysis is the pedigree of failure rate or event probability data that is assigned to basic events. A good faith effort must be made to obtain the best failure rate data that is available. The uncertainty in failure rate data depends in large part on the applicability of the data (its source). A failure rate should apply to the particular application of a component, its operating environment, and its non-operating environment. The failure rate data hierarchy is as follows:

1. Actual mission data on the component,
2. Actual mission data on a component of similar design,
3. Life test or accelerated test data on the component,
4. Life test or accelerated test data on a similar component,
5. Field or test data from the component supplier,
6. Specialized data base or in-house data base on similar components, and
7. Standard handbooks for reliability data.

The component failure probability P, which is also called the unreliability, is determined from the formula

$$P = 1 - e^{-\lambda t}, \tag{7.3}$$

where λ is the component failure rate and t is the relevant time interval. For small values of λt ($\lambda t < 0.1$) the above formula for P simplifies to

$$P \cong \lambda t. \tag{7.4}$$

The units of λ are the failure probability per unit time, e.g. per hour of exposure. For most FT software, the failure rate and time interval can be separately input as data.

Table 7-1 illustrates a sample of component failure rate data used for the Space Shuttle PRA evaluation. The component failure rates are given in the column labeled “Rate” and are in units of per hour. The failure rates are assembled from historical data and from expert inputs. The other columns give attributes of the component failure rate that allow component failure probabilities to be calculated for the appropriated component failures identified in the Shuttle PRA model. Uncertainty information associated with the component failure rate estimates is given in the last three columns for uncertainty propagation.

* If sufficient time-to-failure data from life testing or field operation is available, both random and wearout failures can be accounted for using the Weibull distribution, as described in Appendix C, Section C.4.

Table 7-1. Illustrative Component Failure Rate Data

NAME	DESC	EVENT LEVEL	PART NAME	PART NUMBER	SOURCE	RELATED CIL	LOCATE	SUBSYSTEM	NUMBER OF FAILURES	EXPOSURE	UNITS	RATE	DELTA T	PF MISSION	DIST	UNCERTAINTY PARM1 (MEDIAN)	PARM2 (EF)
042BD0101A	APU 1 BURST DISK FAILS TO BURST	BE	BURST DISK - SEAL CAVITY DRAIN	48-6806, ME251-0017-0001	PEAPU1SCD	04-2-BD01-01	1R/2	04-2	0	0	X	2.55E-05	1	0.0000255	L	6.58E-06	15
042BD0101A	APU 2 BURST DISK FAILS TO BURST	BE	BURST DISK - SEAL CAVITY DRAIN	48-6806, ME251-0017-0001	PEAPU2SCD	04-2-BD01-01	1R/2	04-2	0	0	X	2.55E-05	1	0.0000255	L	6.58E-06	15
042BD0101A	APU 3 BURST DISK FAILS TO BURST	BE	BURST DISK - SEAL CAVITY DRAIN	48-6806, ME251-0017-0001	PEAPU3SCD	04-2-BD01-01	1R/2	04-2	0	0	X	2.55E-05	1	0.0000255	L	6.58E-06	15
042BD0102A	APU 1 BURST DISK EXTERNAL LEAKAGE	BE	BURST DISK - SEAL CAVITY DRAIN	48-6806, ME251-0017-0001	PEAPU1FLK, PEAPU1SCD	04-2-BD01-02	1/1	04-2	0	0	H	2.55E-05	217.5	0.0055463	L	6.58E-06	15
042BD0102A	APU 2 BURST DISK EXTERNAL LEAKAGE	BE	BURST DISK - SEAL CAVITY DRAIN	48-6806, ME251-0017-0001	PEAPU2FLK, PEAPU2SCD	04-2-BD01-02	1/1	04-2	0	0	H	2.55E-05	217.5	0.0055463	L	6.58E-06	15
042BD0102A	APU 3 BURST DISK EXTERNAL LEAKAGE	BE	BURST DISK - SEAL CAVITY DRAIN	48-6806, ME251-0017-0001	PEAPU3FLK, PEAPU3SCD	04-2-BD01-02	1/1	04-2	0	0	H	2.55E-05	217.5	0.0055463	L	6.58E-06	15
042BD0103A	APU 1 BURST DISK INTERNAL LEAKAGE OR PREMATURE RUPTURE	BE	BURST DISK - SEAL CAVITY DRAIN	48-6806, ME251-0017-0001	PEAPU1SCD	04-2-BD01-03	1R/2	04-2	0	0	H	2.55E-05	217.5	0.0055463	L	6.58E-06	15
042BD0103A	APU 2 BURST DISK INTERNAL LEAKAGE OR PREMATURE RUPTURE	BE	BURST DISK - SEAL CAVITY DRAIN	48-6806, ME251-0017-0001	PEAPU2SCD	04-2-BD01-03	1R/2	04-2	0	0	H	2.55E-05	217.5	0.0055463	L	6.58E-06	15

The second type of probability data, an *event occurrence probability* in some time interval, is similar to a component failure probability in some time interval. An event *occurrence rate* and the *time interval* must be supplied. The formulas for the probability of the event occurrence are the same as above, with λ now the event occurrence rate. Event occurrence rates and exposure time intervals are used for fire occurrences, rupture occurrences, and other initiating or occurring events for which there are data on event occurrence rates (e.g., in units of per year or per hour) for the event.

The third type of probability data that may be required is an *unavailability*. This data is supplied for a component that is repairable or checkable. Events in the FT requiring this type of data include cases where a component is out of service and unavailable if called upon to operate. For a component unavailability, a component *failure rate* and *repair time* or *test interval* must be supplied. The specific data required and the associated expression for the component unavailability depends on the type of component. The two usual expressions for the component unavailability "q" are

$$q = \lambda_0\tau/(1+\lambda_0\tau) \cong \lambda_0\tau \quad \text{for an operating component} \tag{7.5}$$

and

$$q = (1/2)\lambda_s T/(1+1/2\ \lambda_s T) + 1 - e^{-\lambda_0 \tau} \cong (1/2)\lambda_s T + \lambda_0\tau \quad \text{for a standby component.} \tag{7.6}$$

In the above formulas, for an operating component λ_0 is the operating failure rate and τ is the average repair time. For a standby component λ_s is the standby component failure rate, T is the test or inspection interval, λ_0 is the operating failure rate and τ is the operating time if the component needs to operate for a time after being called upon. (Note that strictly speaking, q in Equation 7.6 is really an unavailability (the first term) plus an operational failure probability (the second term)). An example of a standby component would be a battery that is called upon to operate if normal electric power is lost. An example of an operating component would be an operating pump that continuously circulates water through a cooling system. For most FT software, the component failure rate(s), e.g., λ_0 and/or λ_s, and the appropriate time intervals, e.g. τ and/or T are input. More detailed expressions for the unavailability can be applied for special situations such as when component testing is staggered. FT software documentation frequently describes these other expressions.

The fourth, and last type of probability data that may be required is a *pure event probability*. A pure event probability is also sometimes called a probability per act or probability per demand. A pure event probability is not decomposed into more basic parameters and can be input for any event. It generally is only input for an event for which a failure rate or occurrence rate per unit time is not recorded. Examples of events for which pure event probabilities are generally input are human errors (input as a probability per demand or per act) and pivotal events, which are generally conditional probabilities. Another example of an event probability would be the probability of a relief valve failing to lift once demanded (i.e., once the system pressure exceeds the lift pressure). Some components have both a pure demand failure rate and a per-hour failure rate. For these components, the demand failure rate needs to be added to the probability of component failure using the per-hour failure rate.

7.2 Data Requirements

To carry out quantifications of an FT, quantitative data need to be input for the basic events. In the simplest form, the input data consists of probabilities for the basic events of the fault tree. As was described in preceding sections, the basic event probabilities can be used in the fault tree equations to determine the probabilities of every higher event in the fault tree, including the top event. Data bases exist that provide basic data for calculating probabilities of different types of basic events. The data in these data bases are used in standard reliability and probability formulas to calculate the basic event probabilities. The use of component failure rates was discussed in a previous section. Available fault tree computer codes use basic input data to calculate the basic event probabilities.

The basic data that are required depends on the type of basic event being quantified. The list below gives the types of data that are generally required:

Component failure rate data. Component failure rates are required to calculate component failure probabilities and component unavailabilities. If a component failure rate data base is used, as is the general case, then the failure mode of the component in the data base should be the same as the failure mode being evaluated in the fault tree. If a component failure rate of the same failure mode is selected from a data base then the specifications of the component in the database and its environment should also match as closely as possible to the component being evaluated. This includes determining that the boundaries of the component are similar (e.g., if

the component defined in the data base includes failure contributions from its power supply and the one being evaluated doesn't, then the data or the model should be modified to insure consistency). Often, the component being evaluated can only be approximated or can only be bracketed by component specifications in the data base. It is therefore important to define associated uncertainty bounds that not only cover statistical uncertainties in the estimate of a given data value, but also the variations in possible applicable data values (this uncertainty is sometimes referred to as the "tolerance" uncertainty and can often be much larger than the statistical uncertainty).

In addition to defining the component specification, it is important to identify whether a standby failure rate or an operating failure rate is required, or both. A standby failure rate is used to determine the probability of a failure to start from a standby condition. An operating failure rate is used to determine the probability of failure to operate for an operating component that has successfully started. For an operating failure rate, a time of operation is also needed to determine the failure probability. For a standby component, the average time that a standby failure exists before repair is accomplished is needed to calculate the unavailability. Failure rates are sometimes given in units or per demand instead of per hour. The demand can be an operational cycle of the component for a cyclic component, an average operation time for an operating component, or an average time a failure exists before completion of repair. The analyst needs to determine which is the definition used for a given data value and if it is applicable, e.g. if the average time that a failure exists is similar to that being evaluated.

Human error data. Data in human error data bases consists of human error rates per action. Human error quantification and human error reliability is different from human factors analysis. Human factors analysis is a psychological assessment of the factors affecting human behavior. Human factors analysis is a qualitative analysis. Human reliability analysis quantifies the probability of different types of human actions. It is human reliability analysis that is used in FTA.

To quantify the human error that is modeled in the fault tree using a human error rate data base, the human error being modeled should be matched as closely as possible to the same type of human error in the data base. Often expert opinion will be used to estimate the human error rate because of the inapplicability of available human error data. In any case, the uncertainty associated with the human error rate should account for the possible variations in human performance and conditions as well as statistical estimation error. Oftentimes, conservatively high human error rates are assigned to determine the resulting sensitivity of the top event probability. Those human error rates for which the top event probability is sensitive are then assessed with more accuracy.

Common cause failure data. Common cause failure (CCF) data are required when CCFs are modeled in the fault tree. The modeling of CCFs was discussed in an earlier section and is addressed in the Sample Container Seal Design Example in Chapter 12. Generally, when CCFs are modeled, conditional failure probabilities are required that are termed beta factors. For more detailed CCF models, more detailed conditional failure probabilities are used that account for the number of previous failures. These more detailed models utilize data that are termed alpha factors, beta binomial factors, or multiple Greek Letter factors. Specialized CCF data sources are available for some components. Often, a conservatively high value is used for the

conditional failure probability for a sensitivity study. If the top event is sensitive to the value then a more accurate assessment is performed.

Phenomenological data. Phenomenological data are needed if one or more initiating event occurrences are modeled as basic events in the fault tree. These phenomenological events can include micrometeoroid or orbital debris (MMOD) damage, a fire or explosion, or an earthquake. The basic data are event frequencies per mission or per unit time. These frequencies, if small, are often used as probabilities and if they are in units of per unit time can be multiplied by the appropriate time interval. For a high value of a frequency, the top event is then the frequency of occurrence (or the expected number of top event occurrences in a given interval). Specialized data bases or expert opinion are used to obtain these estimates along with their uncertainty ranges.

In addition to initiating event probabilities, additional probabilities may be required if sequences are modeled in the fault tree that begin with the phenomenological event. These special probabilities can include the probability that a defect is missed by inspection and the probability that the defect propagates to failure. These probabilities are obtained from external calculations, e.g., stress versus strength calculations, or from expert opinion.

7.3 Top Event Probability

The top event probability is a general term for the probability calculated for the top event. The top event probability is calculated from the fault tree using the probabilities that are input for the basic events. Depending on the specific top event definition, the top event probability can be the probability of the top event occurring during a mission, the probability of the top event occurring in a given period of time, a pure probability number for the top event, or the top event unavailability, i.e., the probability that the top event exists at a given point in time. In some cases, which are not usual, the top event probability can be the frequency of the top event occurring or the expected number of occurrences of the top event in some interval, which can be greater than one. This only occurs if the inputs are basic event frequencies or expected numbers of occurrences that are greater than one in value, which is only done in special cases and is handled by special computer algorithms. The calculation of the top event probability is described in Appendix A.

7.4 Gate Probability

A gate probability is sometimes called an intermediate event probability since it is for an intermediate event below the top event. The gate acts like a top event for the fault tree below it. Thus, everything that is calculable for the top event is also calculable for every intermediate event in the fault tree. The types of probabilities that can be calculated for an intermediate event are the same as those for the top event.

7.5 Importance Measures for a Fault Tree

One of the most important outputs of an FTA is the set of importance measures that are calculated for the top event. These top importance measures establish the significance for all the events in the fault tree in terms of their contributions to the top event probability. Both intermediate events (gate events) as well as basic events can be prioritized according to their

importance. Top importance measures can also be calculated that give the sensitivity of the top event probability to an increase or decrease in the probability of any event in the fault tree. Both absolute and relative importance measures can be calculated.

What is often useful about the top event importances is that they generally show that relatively few events contribute to the top event probability. In many past FTAs, less than 20% of the basic events in the fault tree were important contributors, contributing more than 90% of the top event probability. Moreover, the importances of events in the fault tree generally cluster in groups that differ by orders of magnitude from one another. In these cases, the importances are so dramatically different that they are generally not dependent on the preciseness of the data used in the FTA.

In addition to providing the significance of the contributors, the top importances can be used to allocate resources. These resources might include testing and maintenance resources, inspection resources, upgrade resources, quality control requirements and a wide variety of other resource expenditures. By using the top importances, resources can be optimally adjusted to minimize total resource expenditures while maintaining the top event probability, thus providing a win-win situation. Alternatively, for a given resource expenditure such as for upgrades or for maintenance, the top importances can be used to allocate resources to minimize the top event probability. This aids decision makers in obtaining the "biggest bang for the buck" by providing an objective assessment using systematic methodologies, with associated software if needed, to supplement and complement their subjective information.

These optimizations have been used in various industries to reduce resources by as much as 40% while at the same time maintaining or decreasing the top event probability. An advantage of these optimal allocation approaches is that relative risk importances can be used, which have generally smaller uncertainties than absolute values. Moreover, uncertainties in the importances can also be handled.

In addition to allocating resources, the importances can be used to assign allowed downtimes and repair times (actually a type of resource), to focus diagnostic activities in identifying the causes of a top event, and to focus design activities and requirements in design applications.

Four basic types of top importances can be calculated for the different types of applications that are described above [1]. The basic importance measures that can be calculated for each event in the fault tree are:

Fussell-Vesely (F-V) Importance—the contribution of the event to the top event probability. This importance measure is sometimes call the *Top Contribution Importance*. Both the absolute and the relative F-V importance are determinable for every event modeled in the fault tree, not only for the basic events, but for every higher-level event and contributor as well. This provides a numerical significance of all the fault tree elements and allows them to be prioritized. The F-V importance is calculated by summing all the causes (minimal cut sets) of the top event involving the particular event.

Risk Reduction Worth (RRW)—the decrease in the probability of the top event if a given event is assured not to occur. This importance measure can also be called the *Top Decrease*

Sensitivity. This measure is related to the previous F-V importance. The risk reduction worth for a basic event shows the decrease in the probability of the top event that would be obtained if the lower level event, e.g., the failure, did not occur. It thus gives the maximum reduction in the top probability for the upgrade of an item. Both the absolute value and relative value of the risk reduction worth are determinable for every event and contributor modeled in the fault tree. The risk reduction worth is normally calculated by re-quantifying the fault tree or the minimum cut sets with the probability of the given event set to 0.0. This calculation and that for Risk Achievement Worth and Birnbaum's importance measure (below) are similar to a partial derivative, in that all other event probabilities are held constant.

Risk Achievement Worth (RAW)—the increase in the top event probability if a given event occurs. This importance measure can also be called the *Top Increase Sensitivity*. The risk achievement worth shows where prevention activities should be focused to assure failures don't occur. Since the failures with largest risk achievement worth have the largest system impacts, these are the failures that should be prevented. The risk achievement worth also shows the important events for contingency planning: those events having the largest risk achievement worth have the biggest impacts and should be the priority events considered in contingency plans and reaction plans. Again, both the absolute and relative risk achievement worth are obtainable for every event and contributor modeled in the fault tree. The risk achievement worth is normally calculated by re-quantifying the fault tree or the minimum cut sets with the probability of the given event set 1.0.

Birnbaum's Importance Measure (BM)—the rate of change in the top event probability as a result of the change in the probability of a given event. BM is equivalent to a sensitivity analysis and can be calculated by first calculating the top event probability with the probability of the given event set to 1.0 and then subtracting the top event probability with the probability of the given event set to 0.0. Because of the way BM is formulated, it does not account for the probability of an event. BM is related to RAW and RRW. When these are expressed on an interval scale (absolute value), BM = RAW + RRW.

The above importance and sensitivity measures can be calculated, not only for the fault tree, but also for its equivalent success tree. When applied to the success tree, the measures give the importance of an event not occurring. The top event is now the nonoccurrence of the undesired event and each event is the event nonoccurrence. Therefore, when applied to an event in the success tree, the F-V importance gives the contribution of the nonoccurrence of the event to the nonoccurrence of the top event. The risk reduction worth gives the decrease in the nonoccurrence probability of the top event if the event nonoccurrence probability were zero, i.e., if the event did occur. The risk achievement worth gives the increase in the nonoccurrence probability of the top if the nonoccurrence probability of the event were 1.0, i.e., if the event were assured not to occur. Thus, the importance measures for the success tree give equivalent information as for the fault tree, but from a nonoccurrence, or success, standpoint.

7.6 Uncertainty Analyses in FTA

The data used for FT quantification is uncertain in part because the estimates are obtained from limited sampling (i.e., statistical uncertainty) and in part because the events and/or the way they

are applied varies from the application from which the data was obtained (i.e., tolerance uncertainty). There are two basic approaches for evaluating data uncertainties, *sensitivity analysis* and formal *uncertainty analysis*. Each of these approaches are briefly discussed below. The objective here is to give the reader an understanding of the basic concepts of these approaches as used in FTA. For more details the reader is advised to see any of the many references on sensitivity and uncertainty analysis [2].

In a *sensitivity analysis*, an input data parameter, such as a component failure rate is changed, and the resulting change in the top event probability is determined. This is repeated for a set of changes using either different values for the same parameter or changing different parameters, e.g., changing different failure rates. Usually for a given sensitivity evaluation, only one parameter is changed at a time. This is called a one-at-a-time sensitivity study. Sometimes two or more parameter values are simultaneously changed to study the interactions among the parameters.

The decisions that must be made in carrying out sensitivity analyses include what data parameters to change and what values to use for the changes. A small change in a data parameter shows the linear effect of the change on the top event probability. Assigning a value or 1 or 0 to the failure probability of a basic event provides the maximum effect of changes in the parameter. It can be shown that using a small change and using 1 or 0 for the event probability gives the same information as the top event importance measures that were described in a previous section, i.e., the F-V importance, the risk achievement worth, and the risk reduction worth. Therefore, intermediate changes in the parameters are selected for the sensitivity studies. Some FT software allows a series of systematic changes in the parameters to be carried out, e.g., changing each failure rate by a factor or 2 then a factor of 4, etc.

In contrast to a sensitivity analysis, in an *uncertainty analysis* a probability distribution is assigned to each data parameter to describe the possible values that the data parameter may have. A probability distribution, for example, is assigned to a component failure rate to describe uncertainties in the estimated failure rate value and uncertainties in the application of this failure rate to a particular design. The type of probability distribution and the distribution parameters must be selected to define the specific distribution. The types of probability distributions that are usually used are the *beta* distribution, the *gamma* distribution, and the *log normal* distribution. A histogram is also sometimes used if there are assigned discrete values that the data parameter may assume (with associated probabilities). However, for most FT evaluations one of the parametric distributions are used, i.e., the beta, gamma, or the lognormal. For any distribution, the mean value, median value, standard deviation, 5% lower bound and 95% upper bound give a comprehensive set of characteristic values that summarize the possible values of a data parameter. When one value is reported it is often the mean value. However, the set of characteristic values should be ideally reported to provide a comprehensive summary. Many data bases give at least some of these characteristic values for a given data estimate along with the probability distribution that is used. However it is important to note that these values only address the uncertainty associated with the sampling process, and not the application variation.

The *beta* distribution is used when the data parameter is a pure probability. The possible argument values for the beta distribution range from zero to one. Two distribution parameter values are required to define the specific beta distribution. The 5% lower bound and 95% upper

bound are often used. The mean and standard deviation are also used and are determinable from the lower and upper bounds. These values must be assigned by the user or be extracted from the database being used. Depending on the values of the beta parameters, the beta distribution can be symmetric, can be skewed to the left (a longer tail for lower values), can be skewed to the right (a longer tail for larger values), or can be u-shaped. The beta distribution is used because it allows easy manual updating of the distribution parameters when failure data or event data are collected to give the revised, posterior distribution.

The *gamma* probability distribution is used to describe the uncertainty in component failure rates and occurrence rates per unit time. The possible values for the argument of the gamma range from zero to infinity. The gamma is a positively skewed distribution with a longer tail for larger values. The gamma is a two-parameter distribution and any two characteristic values determine the specific gamma distribution. Usually the 5% lower bound and 95% upper bound or the mean and standard deviation are used to define a specific gamma distribution. These parameters must again be assigned by the user. Like the beta distribution, the gamma distribution is used because it is easy to manually update the gamma parameters for observed failure data or occurrence data to obtain the revised, posterior distribution.

The *lognormal* distribution is the probability distribution most often used to describe uncertainties in data for a FT. The range of the lognormal argument is from zero to infinity. The lognormal is a positively skewed distribution. It is used to describe uncertainties in failure rates and occurrence rates per unit time. It can also be used to describe uncertainties in event probabilities if the values are truncated to be no larger than 1.0. Most FT software uses the lognormal as an uncertainty distribution, and some only have this option. When a lognormal distribution is used this choice implies that on a log scale the data parameter has a standard, normal probability distribution. In using a lognormal, factors are the basic descriptors that define changes in the value, e.g., a factor of 3 change, a factor of 10 change, etc. These “error” factors are natural measures of change to describe uncertainties in FT data. The lognormal is a two parameter distribution and any two characteristics such as the median and 90% (two-sided) error factor define the specific lognormal. The median value divided by the error factor gives the 5% lower bound. The median value multiplied by the error factor gives the 95% upper bound. The mean is simply obtained from the median and error factor, as is the standard deviation. These values must be supplied by the user. Empirical fits that have been carried out show that the lognormal is one of the best fitting distributions to observed data variations.

Once the probability distributions are assigned to each data parameter then the distributions are propagated through the fault tree to determine the probability distribution of the top event. This distribution is interpreted as describing the uncertainty in the top event probability. The mean, median, standard deviation, 5% upper bound, and 95% lower bound are reported for a full summary of uncertainty characteristics. A histogram of the top event probability distribution is also shown and is often fitted to a standard distribution such as a lognormal.

Most FT software determines the top event probability distribution using Monte Carlo simulation. In a Monte Carlo simulation, a value is selected (sampled) from each data parameter distribution. This is done using statistical sampling techniques. Each set of data values are used to quantify the fault tree to determine a value for the top event probability. This is repeated a

large number of times, such as 10,000* or more, to obtain an accurate estimate of the top event probability distribution. The sample of top event probabilities is then used to obtain the top event distribution characteristics and the top event distribution histogram.

Finally, in sampling, the user has the choice of assigning the data distributions to be independent or to be coupled in the sampling. In independent sampling, a value is independently sampled from each probability distribution. In coupled sampling, one probability distribution is used and a value sampled from it and used for multiple data values. Generally, the data distributions should be assigned to be coupled if the same data is used for multiple similar events in the fault tree. This is the case, for example, if the same generic operated motor valve failure rate is used for several motor operated valves in the fault tree. FT software documentation describes the sampling techniques that are available as well as capabilities for coupled and independent sampling.

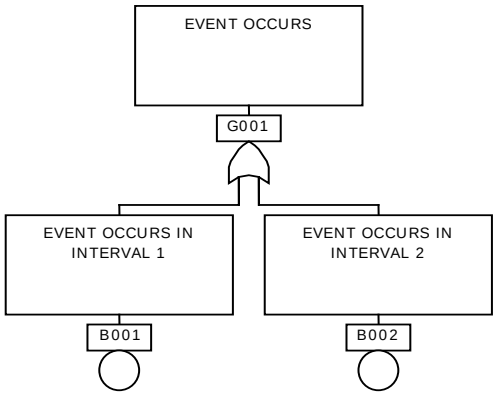
7.7 Phase Dependent and Time Dependent Analyses

A simple fault tree quantification provides one value for the probability of the top event along with the associated uncertainty. This top event probability is not partitioned into contributions over different phases or different time intervals. If the mission under consideration has different phases and these are reflected in the fault tree then the top event probability obtained is the total probability for the mission. Similarly, if a system failure is modeled over a time interval then the top probability obtained is the total system failure probability over the time interval. In this case, individual probabilities for different segments of the time interval are not obtainable. Most FT software cannot produce phase-dependent or time-dependent results. This is not the limitation of the fault tree model itself but a limitation of the available software. Different phase contributions can be modeled in an FT. Also, individual failure rates and time intervals can be provided for each component. However, typical FT software calculates the total probability only and does not have the capability of breaking the probability into more detailed contributions.

This limitation of FT software is not generally a problem because for most applications a total probability is all that is desired. If phase-dependent or time-dependent results are desired then there are two options. Specialized software can be used that has the capability to perform these calculations. Most of the more specialized software use the minimal cut sets of the fault tree as input and then carry out more detailed quantifications using these cut sets. Alternatively, the fault tree model can be modified to allow phase-dependent or time-dependent calculations to be accomplished with standard FT software. Usually it is more resource effective to use specialized software since the fault tree modifications can be both extensive and intensive. However, the fault tree modifications which would be required will be treated briefly here since they can be useful in cases where only a relatively few events are involved. In the following, the term “time-dependent” calculations will be used to cover both “phase-dependent” and “time-interval-dependent” calculations since the techniques are similar for both types of calculations.

* The actual number of samples required can be formally estimated by a number of techniques. However all of these relate to providing enough samples to allow all failure events of interest to occur. One simple rule of thumb is to sample at a factor of 10 greater then the largest denominator given by the lowest point value of interest. For example if the lowest probability were 1×10^{-3}, or 1 in 1000, then 10,000 samples would be chosen. Another technique is to increase the sample size until the results stabilize.

One method of modeling time dependence in the fault tree is to divide (partition) the occurrence of a given basic event into occurrences in smaller time segments. The event is divided into time interval events using an OR gate. This modeling is illustrated below. In this case the event occurrence is divided into two interval occurrences.



In the above model, the basic event occurrence, such as a component failure, has been separated into two more specific events, the event occurrence in Interval 1 or the event occurrence in Interval 2. These two intervals divide the total interval into two smaller sub-intervals. If mission phases were being modeled then the two intervals would be two separate phases. For more intervals or phases there would be more inputs. The OR gate is more correctly a mutually exclusive OR gate since the event cannot occur in both Interval 1 and Interval 2. If the FT software cannot address mutually exclusive OR gates then the simple OR gate can be used provided the minimal cut sets can be scanned to remove any minimal cut sets that contain both of the events.

Data is then be provided for each of the basic events. Specifically, the failure rate or occurrence rate would be provided for each interval, as well as the length of the specific interval. When the minimal cut sets are obtained they are divided into two sets, those containing the Interval 1 event; and those containing the Interval 2 event. The sum of the minimal cut sets in a given set would then be the probability of the top event for that interval (using the standard rare event, or sum of products, approximation).

If there are multiple events in the fault tree that are to be divided into intervals, the same partitioning would be carried for each basic event in the fault tree. When the minimal cut sets are sorted they would be sorted according to the last interval in the minimal cut set. For example, if a minimal cut set contained the basic events “Event A occurs in Interval 1” and “Event B occurs in Interval 2” that cut set would be placed into the set of cut sets for Interval 2. The occurrence of Event A in the earlier interval did not cause the top event. It was the later occurrence of Event B that caused the top event to occur in Interval 2. This same sorting procedure applies to any number of intervals which may be modeled.

The above approach applies the rare event approximation in calculating the probability of an OR event (a union) as the sum of the more specific, input probabilities. As previously discussed, the use of this approximation is not a problem for most applications since it is accurate for most problems of interest. In addition, the above approach is flexible since different failure rates can

be used for each interval, for example to reflect different environments or stresses. The problem with the approach is that it causes the number of basic events on the fault tree to be expanded and hence can greatly expand the number of minimal cut sets that are generated and then must be evaluated.

The number of basic events can be reduced by only expanding particular events and not expanding others. When this is done the top event probability is divided into the intervals in which only particular events have occurred. For other events no differentiation is made as to their interval of occurrence. This approach provides a partial time-dependent answer but even this partial answer can be useful if the focus is on the dominant events or on the events of particular interest. However, for a large fault tree and for many time intervals it is better to use more specialized time dependent software.

Instead of modeling the time dependency in the basic events in the fault tree, separate quantifications can be carried out for each interval. However to allow this to be performed correctly, the FT software must be able to accept initial probabilities for basic events as boundary conditions. In this case, the initial probability for an event is the probability that the event would have occurred prior to the beginning of the calculation. The probability for the first interval is then calculated from the interval length and failure rate for the first interval first. Then the probabilities for each basic event and the top event probability for the first interval are calculated. For the second interval, the first interval basic event probabilities are input along with the failure rates and the interval length for the second interval. This is repeated for each of the intervals in the evaluation. When the minimal cut sets are evaluated for an interval only those contributions for the minimal cut set occurring in the interval are included, i.e., at least one of the basic events in the minimal cut set must occur in the interval for there to be any contribution. This approach can also account for different success criteria or different configurations by using a different fault tree structure in each case. The approach in which repeated calculations are carried out is basically a "Markov" analysis. However, as was mentioned, repeated application of the process can be tedious and most standard FT software does not allow initial probabilities to be provided for events. Therefore, in these cases, it is better to use more specialized software that incorporates the necessary modifications directly as will be addressed in subsequent sections. (The previous approach for modeling the time dependency in the fault tree can also accommodate different fault tree structures for each interval by house events at appropriate points in the fault tree.)

7.8 References

1. W. Vesely et al., *Measures of Risk Importance and their Applications*, NUREG/CR-3385, U.S. Nuclear Regulatory Commission, 1983.

2. T. Bedford and R. Cooke, Probabilistic Risk Analysis: Foundations and Methods, Cambridge University Press, 2001.

8. Dynamic Fault Tree Analysis

In this chapter two special fault tree gates are introduced that make it easier to model systems where the *order* in which events occur affects the outcome. These two special gates are part of the Dynamic Fault Tree (DFT) methodology that has been developed specifically for the analysis of computer-based systems [1]. The concept of *fault coverage* is also introduced, which is used in the fault tolerant computing community to model a phenomenon similar to common-cause failures. The purpose of this chapter is to introduce an interesting extension of the fault tree analysis methodology and to relate concepts used for analysis of computer systems to those used in more traditional FTA.

The DFT methodology was developed to provide a means for combining FTA with Markov analysis. Markov chains [2] are commonly used to assess the reliability and performance of fault tolerant computer-based systems. Markov models have the advantage of easily modeling the sequence-dependent behavior that is typically associated with fault tolerant systems. However Markov models have the disadvantage of being large and cumbersome. The generation of a Markov model for many systems can be tedious and error-prone.

As a simple example of a sequence dependent failure, consider a system with one active component and one standby spare connected with a switch controller [3], shown in Figure 8-1.

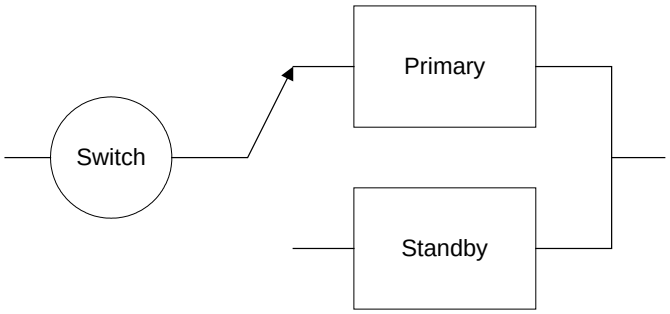


Figure 8-1. Example of Sequence Dependent Failure

Suppose that the failure mode of the switch is such that when it fails, it is unable to switch. The failure of the switch only matters if switching from the primary to the standby spare is of interest. If the switch controller fails *after* the primary unit fails (and thus the standby spare is already in use), then the system can continue to operate. However, if the switch controller fails *before* the primary unit fails, then the standby spare unit cannot be switched into active operation and the system fails when the primary unit fails. The order in which the primary and switch fail determines whether the system continues to operate.

A fault tree model for the standby system is shown in Figure 8-2, where a Priority AND gate captures the sequence dependent failure of the primary and switch. As discussed in Section 4.2, Priority AND gate, the Priority AND gate can be replaced with an AND gate, if the sequence condition is stated explicitly. Thus the fault tree in Figure 8-2 can be replaced with the one in Figure 8-3. To use the fault tree in Figure 8-3, the analyst must calculate the probability associated with the event "Switch fails before primary." In some cases this calculation is trivial. For example, if the time to failure distributions are exponential, and the primary fails at rate λp

while the switch fails at rate λ_S, then the probability that the switch fails before the primary is simply $\lambda_S/(\lambda_P+\lambda_S)$.

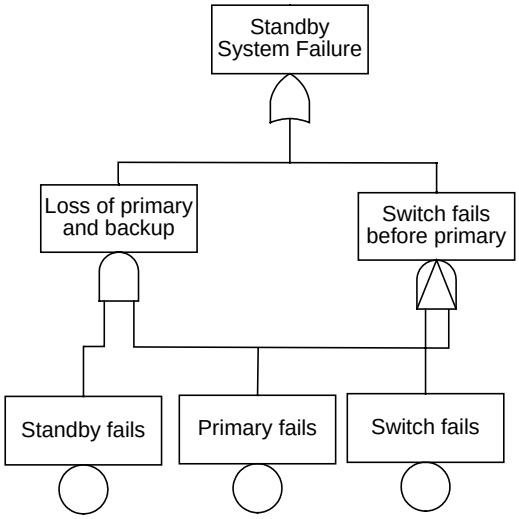


Figure 8-2. Fault Tree Model for Standby System

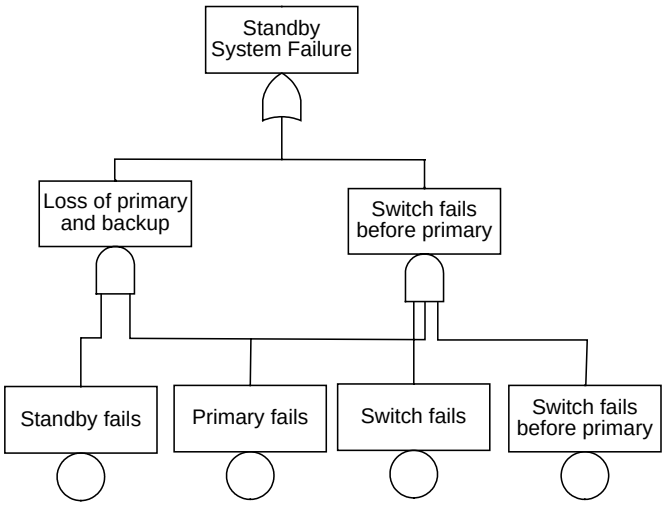


Figure 8-3. Alternate Fault Tree for Standby System

If the inputs to a Priority AND gate are not simply basic events, this calculation is more difficult, and in general requires the solution of a set of integral equations or some approximations [4].

The DFT methodology can relieve the analyst of the need to perform the calculation of the probability of the order of occurrence of the inputs to a Priority AND gate. The DFT methodology automatically generates and solves the set of integral equations needed to solve the Priority AND system.

Although many systems with sequence dependencies can be analyzed using traditional fault tree analysis, the use of DFT constructs offers an interesting alternative. The DFT methodology, in addition to supporting exact solution of the Priority AND gate, facilitates the analysis of other sequence dependencies.

8.1 Sequence Dependent Events and Gates

Functional Dependency

Suppose that a system is configured such that the occurrence of some event (call it a *trigger* event) causes other dependent components to become inaccessible or unusable. For example, a communication network failure could disconnect processing elements, or a power failure could disable other components. A *functional dependency* (FDEP) gate* can be used to explicitly reflect such dependence. An FDEP gate (see Figure 8-4) has a single trigger input (either a basic event or the output of another gate in the tree), and one or more dependent basic events.

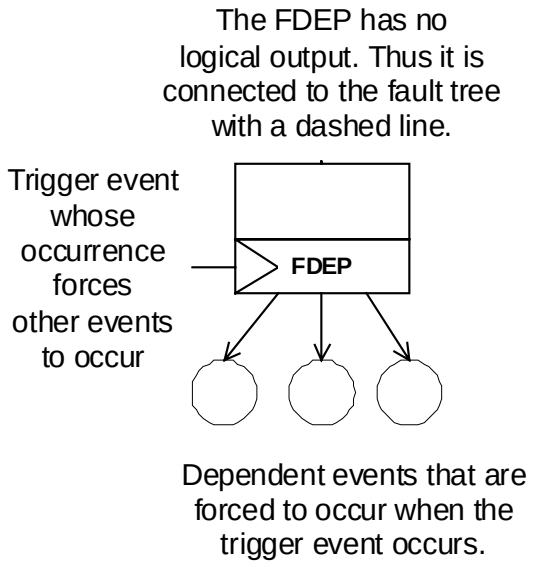


Figure 8-4. Functional Dependency Gate

The dependent basic events are functionally dependent on the trigger event. When the trigger event occurs, the dependent basic events are forced to occur (fail). The separate occurrence of any of the dependent basic events has no effect on the trigger event.

The functional dependency gate is useful, for example, when communication is achieved through some network interface elements, where the failure of the network element isolates the connected components. Figure 8-5 shows the use of an FDEP gate to model network failure that isolates a set of processing nodes. In this case, the failure of the network element is the trigger event and the connected components are the dependent events. When the network element fails, the three processing nodes are set to failed.

* The speciality gates described in this Chapter are a part of the Galileo/ASSAP software package (See Chapter 1, Section 1.7).

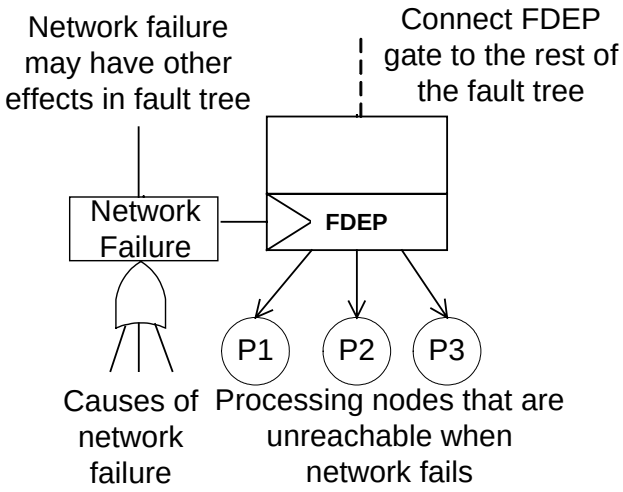


Figure 8-5. Using an FDEP to Model Network Failures.

An alternative to using the FDEP gate in this case is to include network failure as a potential cause of failure for each processing node individually, as in Figure 8-6. That is, the network failure event could input to a set of OR gates, one for each processing node. These two approaches are equivalent as long as the processing nodes do not experience uncovered failures (uncovered failures are discussed later in this chapter).

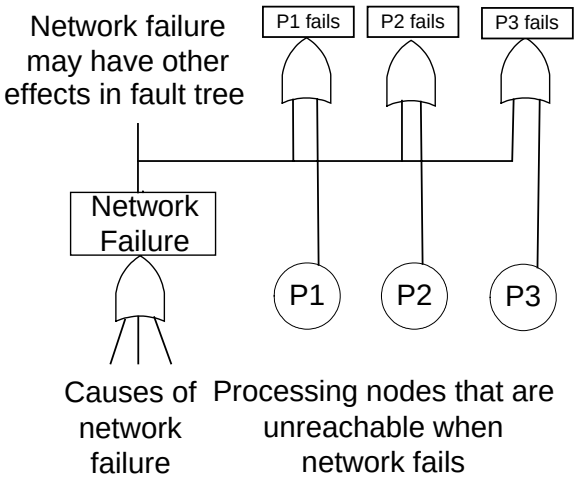


Figure 8-6. Using Standard Fault Tree Constructs to Model Functional Dependence

An FDEP gate can also be used to model systems with interdependencies, which would otherwise introduce loops in the fault tree. In Chapter 5, Section 5.4, it was noted that feedback loops must be avoided in fault tree analysis. This requires only the internal failures in a system providing support be addressed, which often requires multiple fault trees to be developed for a support system. However, the use of the dynamic FDEP gate allows such loops to be modeled correctly. Suppose for example that thermal control and power systems are interconnected, such that each needs the other to function. A pair of FDEP gates can reflect this interdependence.

Figure 8-7 shows that the failure of the thermal system affects the power system, and that the failure of the power system affects the thermal system.

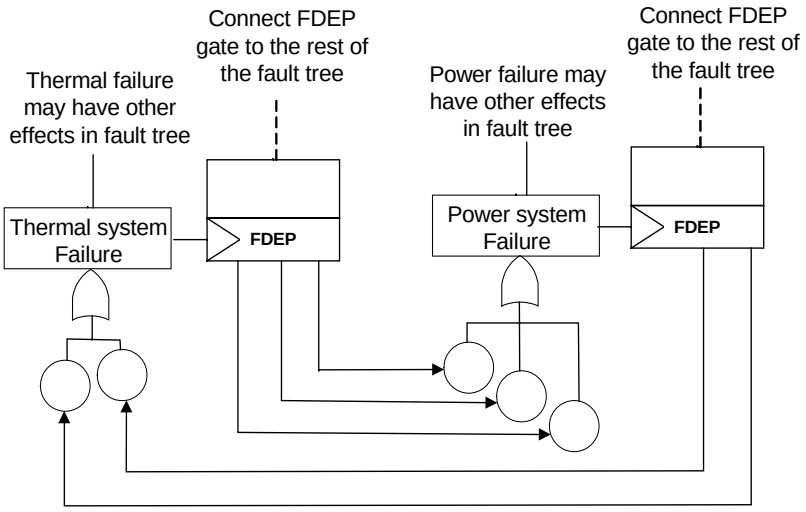


Figure 8-7. Accounting for Feedback Using FDEP

Spare Gate

Consider a system that utilizes cold spares (spare components that are unpowered, and thus do not fail before being used). Such systems can be difficult to model exactly using standard fault tree techniques. The difficulty arises because the system failure criteria cannot be expressed in terms of logical combinations of basic events, all using the same time frame.

Figure 8-8 shows a SPARE gate. The inputs to the SPARE gate are all basic events and are ordered. The first (usually drawn as leftmost) input is the primary event, while the second and subsequent inputs represent spares. The SPARE gate models the sequential activation of the spares: the first spare is activated when the primary fails; the second when the first fails, etc. The SPARE gate has one output that becomes true after all the input events occur.

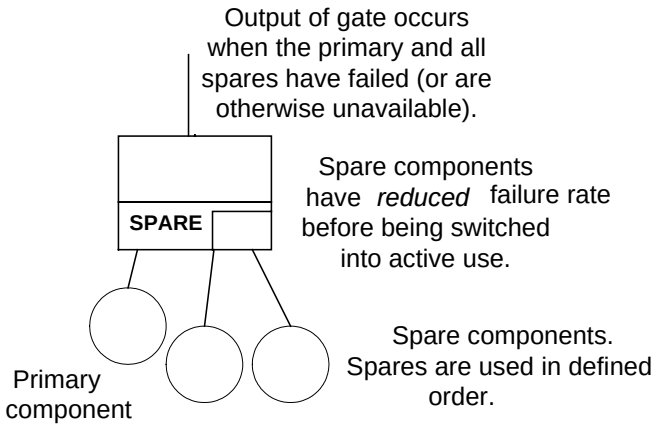


Figure 8-8. Spare Gate

Associated with each input to the SPARE gate is a dormancy factor (usually between zero and one) that multiplies the failure rate while the unit is spare. Thus, it can be assumed that spares are cold, warm or hot. Cold spares do not fail (if dormancy is zero). Hot spares fail at the same rate as active (if the dormancy is one). Warm spares fall somewhere between cold and hot spares.

The DFT methodology keeps track of the order of occurrence of events in the fault tree by automatically generating and solving the equivalent Markov model. Markov modeling is described in Appendix D.

The SPARE gate can be used if spare units are shared, that is, where a pool of spares is available to replace any of a set of (presumably identical) components. A basic event representing a shared can be an input to more than one SPARE gate. In this situation, the spare is only available to one of the SPARE gates, depending on which of the primary units fails first. For example, Figure 8-9 shows a single cold spare A that is available to replace either A1 or A2, whichever fails first.

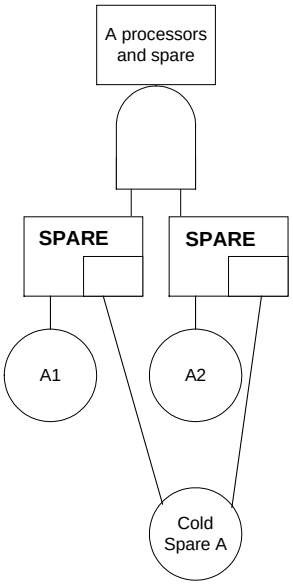


Figure 8-9. Sharing a Spare

8.2 Coverage Modeling for Fault Tolerant Computer-based systems

Within the fault tolerant computing community, the concept of fault *coverage* has been studied, as well as the importance of coverage to accurate reliability assessment [5]. Coverage is defined as the probability that the computer system can recover, given that a fault occurs. Fault coverage is a measure of the computer system's ability to perform fault location, fault containment, or fault recovery [2], [6], [7], [8].

This section introduces the concept of coverage as it is understood in the fault tolerant computing community, and relates it to the concept of common-cause failure, as it is understood in the PRA community.

The concept of coverage in fault tolerant computing

Technological advances have reduced costs to the point where fault tolerant computer system designs can include sufficient redundant components to minimize the probability that all have failed. However, the system designer must be certain that faults and errors are detected promptly, so that the redundant units can be used effectively. If a faulty unit is not reconfigured out of the system it can produce incorrect results that contaminate the non-faulty units. Reliability models of fault tolerant computer systems incorporate *coverage factors* to reflect the ability of the system to automatically recover from the occurrence of a fault during operation. A coverage factor is a probability that the system can automatically recover from a fault and it's associated errors, and thus can continue normal operation, though possibly in a degraded mode.

A fault tolerant computer system may fail to recover from a fault even if spare units remain. For example, a fault may produce an undetected error and the subsequent calculations or operations then operate on incorrect data, possibly leading to overall system failure. Even if an error is detected, the system may still be unable to recover, because the fault could "confuse" the automatic recovery procedures into disabling the wrong component. A *coverage model* is used to help structure our discussion of covered and uncovered faults.

General structure of a coverage model

Figure 8-10 shows the general structure of a coverage model. The entry point to the model is the occurrence of the fault, and the three exits (**R, C**, and **S**) are the three possible outcomes. The **R** or **C** exit is reached when a fault is covered; the **S** exit is reached when a fault is uncovered.

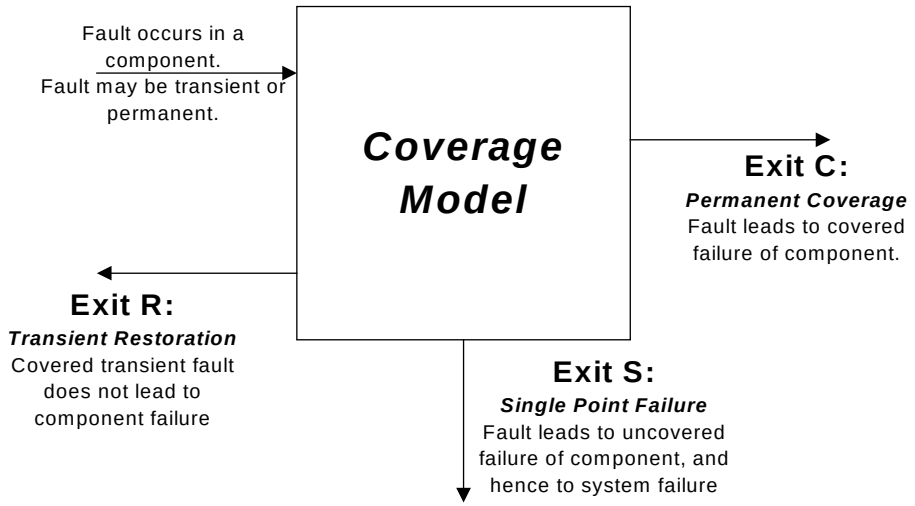


Figure 8-10. General Structure for a Coverage Model

Exit R from the coverage model represents *transient restoration*, the correct recognition of and recovery from a transient fault. A transient may be caused by external or environmental factors, such as excessive heat or a glitch in the power line, or more likely, subtle software defects. The vast majority of faults that occur in computer systems are transient. Successful recovery from a transient fault restores the system to an operational state without discarding any components - for

example by masking the error, retrying an instruction, or rolling back to a previous checkpoint. Reaching this exit successfully requires timely detection of an error produced by the fault; performance of an effective recovery procedure; and swift disappearance of the fault (the cause of the error).

Exit C from the coverage model represents *permanent coverage*, the determination of the permanent nature of the fault, and the successful isolation and (logical) removal of the faulty component.

Exit S from the coverage model represents *single point failure*, in that a single fault causes the system to fail, generally when an undetected error propagates through the system, or if the faulty unit cannot be isolated and the system cannot be reconfigured.

In a reliability analysis of a fault tolerant computer system, each component in the computer system has an associated set of coverage factors (*r*, *c* and s) that represent the probability of reaching the associated exit of the coverage model when a fault occurs in that component. The three exits from the coverage model are mutually exclusive and complete, thus the three probabilities sum to unity. The *s* probability reflects the extent to which a component fault can cripple the system. The *r* and *c* probabilities reflect the extent to which the computer system can automatically recover from a fault. The relative values of *r* and *c* reflect the relative proportion of transient and permanent faults expected to occur.

Incorporating the coverage probabilities into the fault tree model

To illustrate how the coverage probabilities (*r*, *c*, and *s*) for each component are integrated into the solution of a static fault tree model, first consider a simple computer system. The 3P2M system consists of 3 processors (of which one is needed), 2 memories (of which one is needed) connected by a bus. The fault tree model for the 3P2M system is shown in Figure 8-11. The *k* * *X* notation in a basic event represents *k* identical components of type X. This shorthand notation is supported by the DFT methodology as a convenience to the analyst. Suppose that we have defined coverage models for the processors and memories. Such coverage models would include an analysis of the proportion of transient to permanent faults, error detection, recovery mechanisms, and automatic reconfiguration. Figure 8-12 shows conceptually how coverage models are added to the fault tree. The pictures inside the boxes are representations of the coverage models for the components.

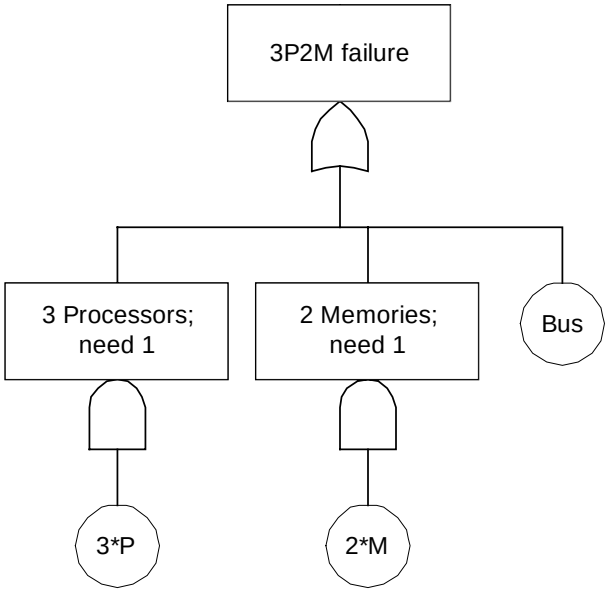


Figure 8-11. 3P2M Fault Tree

In Figure 8-12, the arrow from a basic event to the coverage model represents the occurrence of a fault. The effects of the fault (transient restoration, covered fault or uncovered fault) are the exits of the coverage model. Transient faults (from which the system can automatically recover) lead back to the basic event, representing a fault that had no permanent consequences. A covered fault leads to the normal fault tree gate.

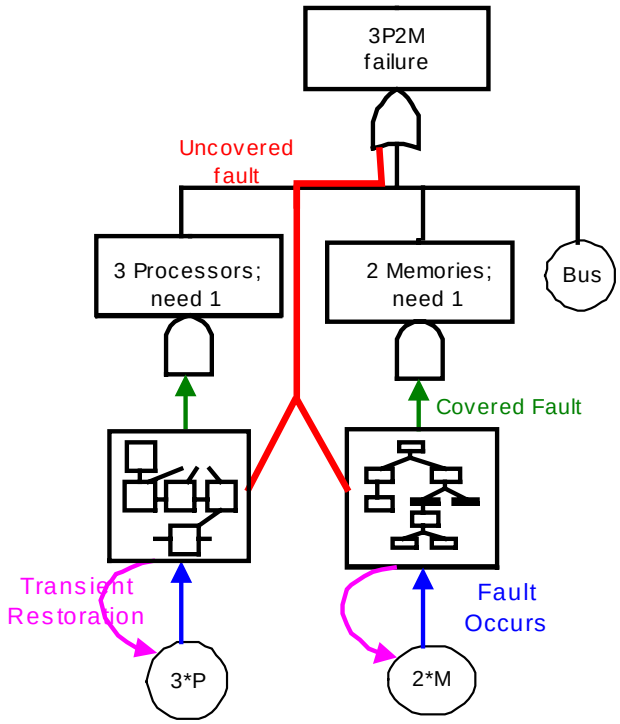


Figure 8-12. Inserting Coverage Model in a Fault Tree

It is covered faults whose combination is represented by the original fault tree model. Uncovered faults lead directly to the top OR gate in the tree. For details on how the coverage probabilities are incorporated into the quantitative analysis of the fault tree, see [9].

Relationship between coverage and common-cause failures

The concept of coverage and the concept of common-cause failures were developed independently to address similar needs in different analysis communities. Probabilistic analysis of redundant systems can produce wildly optimistic results if the redundant components are considered to fail independently.

Common cause failures (CCFs) are described in Chapter 5, Section 5.2, where the concept of a β-factor is introduced. When modeling CCF, one defines a set of (usually identical) components or subsystems, and associated with this set is a β-factor to represent the fraction of the failure rate of a single component that threatens the other components in the set. Suppose that there are three components of type A, with individual failure rate λ_A and a β-factor of β_A. The rate of a common cause failure for all components of type A is then. A CCF affects all components in the common cause set of like components.

Imperfect coverage, or an uncovered fault in a computer system, similarly affects more than a single component. An uncovered fault in component A with individual failure rate λ_A and single-point failure (uncoverage) probability s_A affects the entire computer system with rate $\lambda_A s_A$.

Thus the notions of common-cause and uncovered failures are similar. The main difference is that CCF affect components of the same type, and do not necessarily lead to system failure. Uncovered faults in a computer system can (and usually do) affect components of a different type and lead to the failure of the entire computer system (and presumably the system being controlled by the computer). That is, an uncovered failure in a processor could affect all the components connected to the bus, whether they are processors, memories, devices, etc. Further, coverage modeling allows the distinction between transient faults (which are very common in computer systems) and permanent faults.

8.3 Modular Solution of Dynamic Fault Tree

The use of a Markov model to solve DFTs offers a significant advantage. The Markov model permits failure effects that depend on the order in which components fail to be captured in the analysis. This issue is especially important when modeling systems with shared pools of spares. However, this ability to model complex redundancy management does not come easily. The solution of a Markov model is much more time and memory consuming than the solution of a standard (static) fault tree model. The size of a Markov model (in terms of the number of states and transitions) grows exponentially with the number of components in the system. For a system with many components, the solution of a system using a Markov model may be infeasible, even if the model is truncated.

However, if the system level fault tree can be divided into independent modules [10], and the modules solved separately, then the separate results can be combined to achieve a complete

analysis. As an example, consider the 3P2M example, but consider that one of the processors is a cold spare. The modified 3P2M with fault tree is shown in Figure 8-13. The Markov model for this fault tree has about 30 states, and is readily solved. However, the fault tree may be divided into two modules, as shown in Figure 8-14, where the independent module is indicated. This independent dynamic module can be solved as a Markov model with only 6 states. When the Markov model is solved, its probability of failure is incorporated into the rest of the fault tree, which is now static (it has no dynamic gates), and can be solved via conventional fault tree analysis techniques.

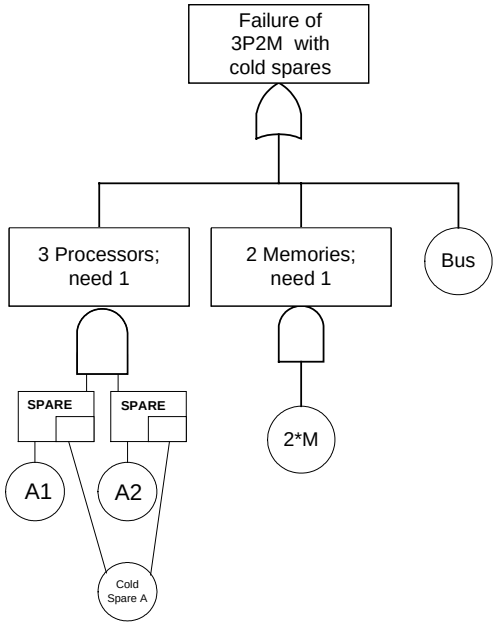


Figure 8-13. Fault Tree for 3P2M System with Cold Spares

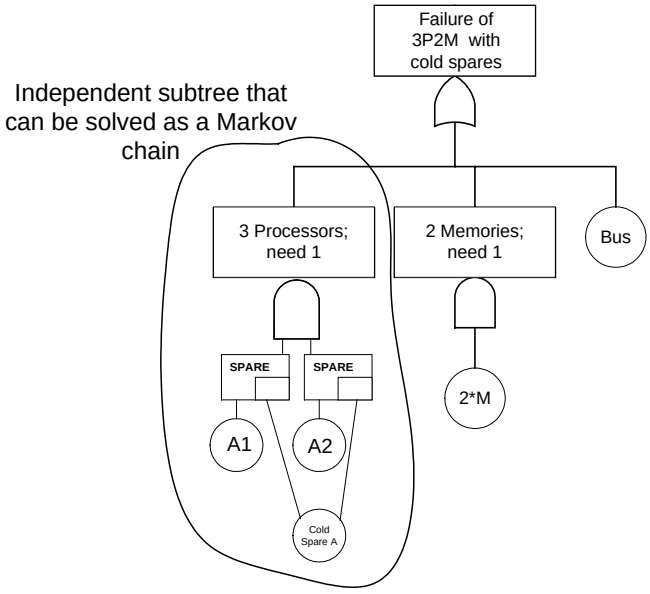


Figure 8-14. Independent Module in Fault Tree

This modularized approach to fault tree analysis has several important advantages. First, it allows for the use of the more efficient static fault tree analysis methodology where it applies, and the Markov approach is used only where necessary. Further, even if every module is dynamic, separate Markov models can be developed for each module, which can result in an enormous saving of computer time and memory. Consider the case of three dynamic modules, each needing 1000 states in the Markov chain. The solution of three 1000-state Markov chains is not computationally intensive. However, without modularization, the resulting single Markov chain would consist of the cross-product of the states of the separate models; that is, a $1000 \times 1000 \times 1000$ states, which is beyond the capabilities of most computer systems.

8.4 References

1. J. Dugan, S. Bavuso and M. Boyd, "Dynamic fault tree models for fault tolerant computer systems," IEEE Transactions on Reliability, Vol. 41, No. 3, September 1992, pp 363-377.

2. J. Pukite and P. Pukite, *Modeling for Reliability Analysis*, IEEE Press, 1998.

3. E. Henley and H. Kumamoto, *Probabilistic Risk Assessment*, IEEE Press, 1992.

4. J. Fussell, E. Aber and R. Rahl, "On the quantitative analysis of Priority-AND failure logic," IEEE Transactions on Reliability, Vol. R-25, No. 5, December 1976, pp 324-326.

5. W. Bouricius, "Reliability modeling for fault-tolerant computers," IEEE Transactions on Computers, Vol. 20, No. 11, November 1971.

6. J. Dugan and K. Trivedi, "Coverage modeling for dependability analysis of fault tolerant computer systems," IEEE Transactions on Computers, Vol. 38, No. 6, pp 775-787, 1989.

7. M. Cukier, J. Arlat and D. Powell, "Coverage estimation methods for stratified fault-injection," IEEE Transactions on Computers, Vol. 48, No. 7, pp 707-723, July 1999.

8. L. Kaufman, B. Johnson and J. Dugan, "Coverage estimation using statistics of the extremes when testing reveals no failures," IEEE Transactions on Computers, Vol. 51, No.1, pp 3-12, January 2002.

9. S. Amari, J. Dugan and R. Misra, "A separable method for incorporating imperfect fault coverage into combinatorial models," IEEE Transactions on Reliability, September 1999, pp 267-274.

10. Y. Dutuit and A. Rauzy, "A linear-time algorithm to find modules in fault trees," IEEE Transactions on Reliability, September 1996.

9. FTA in Aerospace PRA Applications

The following sections discuss FTA topics that are relevant to aerospace applications. Figure 9-1, extracted from the NASA PRA Procedures Guide [1], shows the role of the FT modeling in a typical PRA. The block labeled "Logic Modeling" corresponds to event tree and fault tree modeling of accident sequences (accident scenarios) as discussed previously in Section 1.6 of this handbook. Because FTs are workhorses of a PRA, and can be used as stand-alone models, this handbook focuses on fault trees modeling techniques. For information on scenario-based PRA modeling, the reader should consult Reference 1.

Certain PRA model considerations for aerospace applications are discussed in Section 9.1. The last two sections of the chapter describe two specific uses of FTs either in a PRA or as a stand-alone —to support development of a new design (an important current use by NASA) and to support analysis for an implemented design such as the Space Shuttle.

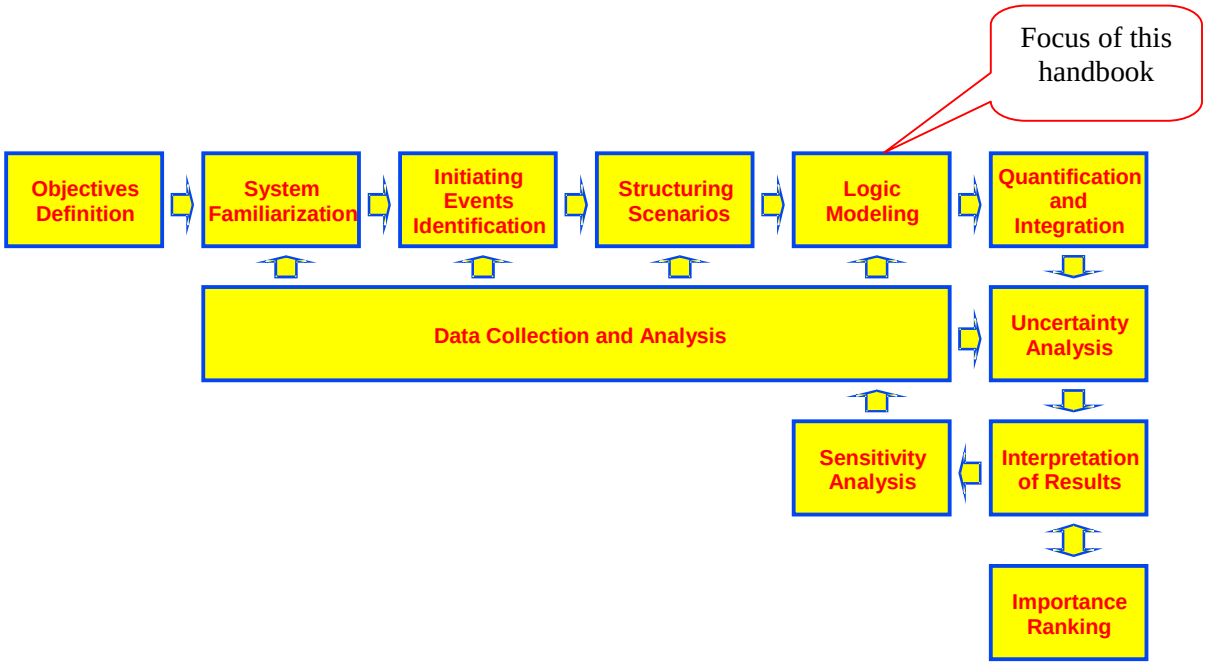


Figure 9-1. A Typical PRA Task Flow

9.1 Separating Qualitative and Quantitative Considerations in FTA as Exemplified in a Phased Mission Analysis

For certain aerospace applications, the goal is to model the different phases of a mission. An example is the Space Shuttle, which can be modeled as having three phases in its mission—Ascent, Orbit and Entry. If a system goes through different phases in a mission then the failure of the system in each phase should be modeled. The success criteria for the system as well as the system configuration and system boundary may change from phase to phase. For each phase a fault tree can be constructed for system failure accounting for the success criteria, configuration, and system boundary for that phase. In constructing the fault tree for the given phase other

phases can be ignored. However, when the fault trees are quantified for the different phases then the interactions among the different phases need to be taken into account. For example, if there is an event in the fault tree that describes the component being failed in a given phase, then the component may be failed due to its failing in the given phase or may be failed due to its failing in a previous phase. When the probability of the component being failed in a phase is quantified then the probabilities of the component failing in the past phases need to be evaluated as well as in the present phase. In this quantification, different component failure rates in the different phases may be used as well as different repair criteria. A number of computer codes handle these inter-phase evaluations to simplify the analyst's tasks. This separation of the qualitative and quantitative considerations in phased missions is an example of the general separation of the qualitative and quantitative considerations in FTA in general.

9.2 Fault Trees for System Design

A fault tree can also be constructed for a system that is being designed as well as for a system that is implemented and operating. Even though the general principles used in constructing these two different types of fault trees are the same, there are differences in the strategies used, in the scope of the fault trees, and the level of resolution of the fault trees. The basic principles applicable to the construction of fault trees for design are discussed here. Fault trees constructed for already operating systems are discussed in the next section.

In constructing a fault tree for a system that is being designed, the detailed specifications for the system or the detailed schematic for the system are not generally available. Often only the top-level logic for the system is available, this consisting of its basic functions and general interfaces. Even with this limited information, a fault tree can be an important tool in assisting in the design of the system. Furthermore, a fault tree can serve as a primary tool for providing a performance-based design for the system.

For evaluating a system design, the fault tree to be developed is a top-level fault tree showing the general logic and relationships for the design. To quantify the fault tree, where specific data are not available, generic data from published data sources are used. When using generic databases, data on heritage systems, suitably modified to take into account risk significant design changes, are used and the data that brackets the component or subsystem that is being investigated is determined by comparing the general characteristics of the design with the characteristics associated with the generic data. The *bracketed* results from the design fault tree gives useful information on the range of failure probability or reliability achievable with the design. One of the example fault trees that will be described was used for a system design.

When the design fault tree is quantified the importances and sensitivities of the different parts of the design are obtained. This is useful information and shows what parts of the system drive the failure probability and reliability. The designer can then focus on the important and sensitive parts. One of the greatest benefits resulting from carrying out any FTA is the establishment of design priorities for all elements of the fault tree and thereby for the design effort. Often, only a few of the elements, or contributors, will drive the failure probability and reliability. The FTAs that have been performed in the past generally show that less than 20% of the contributors dominate the failure probability and the reliability. Often, in fact, 90% or more of the result is driven by as little as 10% or less of the contributors.

The application of the design fault tree can be carried one step further. In this case it can be used as a tool for *performance-based* design. The example of the design fault tree that will be described is an application of FTA for performance-based design. In carrying out a performance-based fault tree evaluation, a failure probability goal is assigned to the top event. This goal is then allocated down through the fault tree to the modules and subsystems in the design. The allocated values that are obtained for the modules and subsystems indicate whether the design has the capability of meeting the top goal. In other words, these values indicate what is sometimes called the "achievability" of the design goal. Various allocations can be selected to determine their feasibilities. Furthermore, by incorporating CCFs into the fault tree evaluations, not only can the number of redundant elements required be determined but also whether diversity is required as opposed to simple identical unit redundancy. For diversity, the redundant capabilities must be provided without relying on identical units to guard against common dependencies. It can be further required that proven technologies be used to provide the functions. In this case, the allocated values provide performance requirements to the suppliers of the system. Design fault trees can therefore be important tools to assist in focusing the design effort and providing performance-based requirements for the design.

9.3 Fault Trees for an Implemented System

When a fault tree is constructed for an implemented and operational system, detailed design and operational information is generally available. In this case, the goal in carrying out a FTA is often to *improve* the system or to *diagnose* problems within the system. The fault tree may also be constructed to *monitor* system safety or reliability performance. When a fault tree is constructed for an implemented system the tree is developed down to a level containing the contributors of interest and for which data is available. This often means constructing the fault tree down to the major component level, e.g. to a valve, pump, and control module level. Because of their low failure probabilities, piping and wiring is not generally modeled unless the objective is specifically to go to this level of detail or if there is suspicion that global effects, such as aging or wearout, have increased the failure probabilities. Also, fault trees are generally not developed to a detailed contact or part level for a component such as a valve stem because of the lack of data to support quantification of such detailed models.

The capability of a FTA to establish priorities among fault tree elements is very useful. Different importance measures may be calculated in FTA for different applications. By including resource expenditures, burden-to-importance ratios can be calculated to show the imbalances between resource expenditures and the importance to risk. Using these importance measures and using cost-benefit principles, resource expenditures on operational systems can be optimally re-allocated to maximize their effectiveness. In many past applications, resources have been re-allocated resulting in significant reductions in total resource expenditures with no impacts on the failure probability or risk. In particular cases, resource expenditures have been reduced as much as 40% or more. If the total resource expenditure is held fixed then the resources can be re-allocated to significantly reduce the current failure probability and risk. In many cases the failure probabilities have been significantly reduced, sometimes by a factor of 10 or more, using the same total resources.

In addition to prioritizing the contributors, FTA can be used on an operational system to predict and correct failures before they occur. Failure trending of the components or other lower level

elements in a system can be established and input to the fault tree to determine the system implications. Performance criteria on the system can then be used to determine the appropriate actions to take. Monitoring of system performance can also be conducted by periodically updating the fault tree quantification with current data. Formal approaches exist for incorporating new data into the baseline fault tree quantification. Defect data and soft failure (partial failures) data can also be incorporated, in addition to hard failure data. The use of FTA in this way can be referred to as a "proactive" use.

The FTA can also be used reactively. In this case, when a system failure occurs the fault tree can be used to diagnose the potential causes and to identify the most effective corrective measures. If a component failure occurs, the fault tree can be used to identify the significance of the failure with respect to the overall failure of the systems and identify those remaining components in the system that are most important in preventing the top event (precursor or "near miss" analysis). This evaluation can sometimes be performed using the importance measures produced by the FTA.

9.4 References

1. *Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners,* NASA, Version 1.1, August 2002.

10. Pressure Tank Example

In this and the next three chapters, undesired events associated with four simple systems are defined and the corresponding fault trees are developed with the help of the rules described in Chapter 4. The fault trees are solved in two of the examples to identify the associated minimal cut sets and limited observations are drawn. The first example, the pressure tank, is a classic fault tree example.

10.1 Pressure Tank System Definition and Fault Tree Construction

Consider Figure 10-1, which shows a pressure tank-pump-motor device and its associated control system that is intended to maintain the tank in a filled and pressurized condition. The operational modes are given in Figure 10-2.

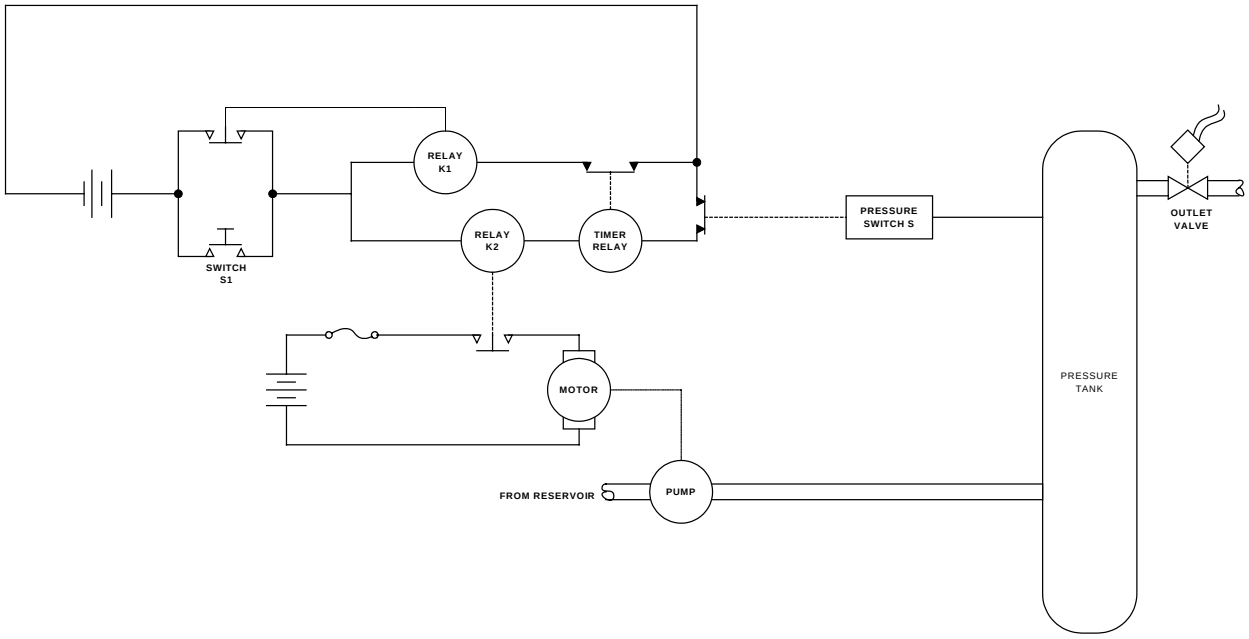


Figure 10-1. Pressure Tank System

The function of the control system is to regulate the operation of the pump. The latter pumps fluid from an infinitely large reservoir into the tank. Assume that it takes 60 seconds to pressurize the tank. The pressure switch has contacts that are closed when the tank is empty. When the threshold pressure has been reached, the pressure switch contacts open, deenergizing the coil of relay K2 so that relay K2 contacts open, removing power from the pump, causing the pump motor to cease operation. The tank is fitted with an outlet valve that drains the entire tank in an essentially negligible time; the outlet valve, however, is not a pressure relief valve. When the tank is empty, the pressure switch contacts close, and the cycle is repeated.

Initially the system is considered to be in its dormant mode: switch S1 contacts open, relay K1 contacts open, and relay K2 contacts open; i.e., the control system is de-energized. In this de-energized state the contacts of the timer relay are closed. The tank is also assumed to be empty and the pressure switch contacts are therefore closed.

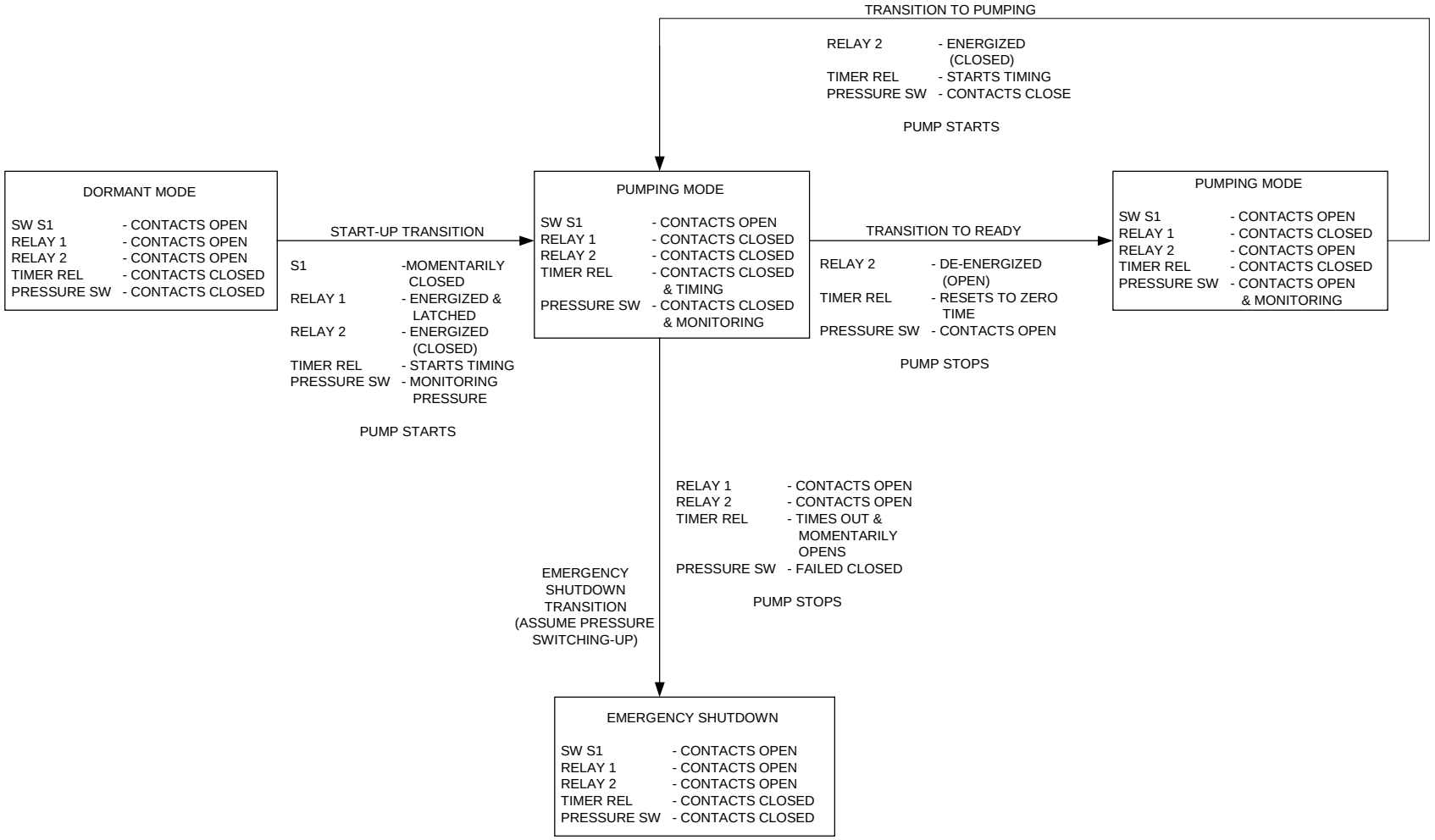


Figure 10-2. Pressure Tank Operational Modes

System operation is started by momentarily depressing switch S1. This applies power to the coil of relay K1, thus closing relay K1contacts. Relay K1 is now electrically self-latched. The closure of relay K1 contacts allows power to be applied to the coil of relay K2, whose contacts close to start the pump motor.

The timer relay has been provided to allow emergency shut-down in the event that the pressure switch falls closed. Initially the timer relay contacts are closed and the timer relay coil is de-energized. Power is applied to the timer coil as soon as relay K1 contacts are closed. This starts a clock in the timer. If the clock registers 60 seconds of *continuous* power application to the timer relay coil, the timer relay contacts open (and latch in that position), breaking the circuit to the K1 relay coil (previously latched closed) and thus producing system shut-down. In normal operation, when the pressure switch contacts open (and consequently relay K2 contacts open), the timer resets to 0 seconds.

The undesired event for this example is:

RUPTURE OF PRESSURE TANK AFTER THE START OF PUMPING

It will simplify things considerably if plumbing and wiring failures and also all secondary failures are neglected except, of course, the one of principal interest: "tank rupture after the start of pumping."

The reader may object that a system that includes an infinitely large reservoir and an outlet valve that drains the tank in a negligible time is unrealistic; and that furthermore plumbing and wiring faults that might contribute to the occurrence of the top event have been neglected. The point is that with this simplified system most of the important steps in fault tree construction can be illustrated. In a more complex system the reader might tend to lose sight of the overall system and become too involved in the details.

First, check to make certain that the top event is written as a fault and that it specifies a "what" and a "when." Next, apply the test question: "Can this fault consist of a component failure?" Because the answer is "Yes," immediately add an OR-gate beneath the top event and consider primary, secondary, and command modes.* The tree has now been developed to the point shown in Figure 10-3.

In this problem the limit of resolution will be established at the "component failure level." "Component" refers to those items specifically named in Figure 10-1. Thus the primary failure of the tank (e.g., a fatigue failure of the tank wall) is already at the limit of resolution and is shown in a circle. Whether or not the statement in the diamond is included is moot. It could be assumed at the beginning that the tank was an appropriate one for the operating pressures involved. At any rate, it is decided not to trace this fault any further.

* In this case, however, there is no command mode.

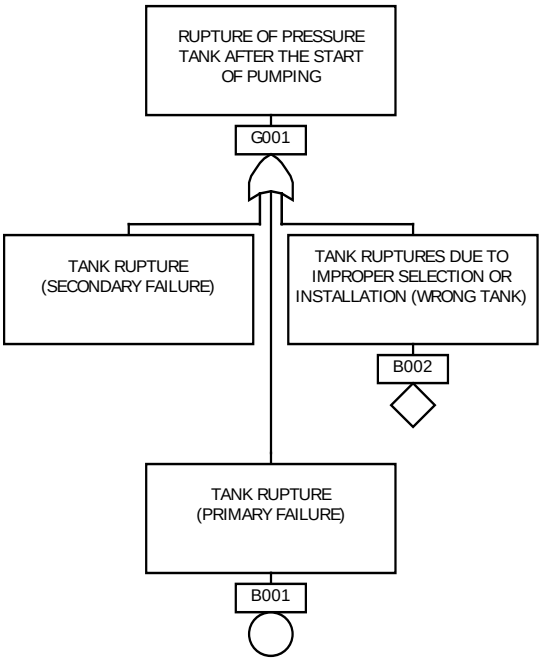


Figure 10-3. Fault Tree Construction – Step 1

Thus, attention is now directed to the secondary failure of the tank. The reader will remember from Chapter 4, that in contrast to a primary failure, which is the failure of a component in an environment for which it is qualified, a secondary failure is the failure of a component in an environment for which it is *not* qualified. Because the secondary failure of the tank can consist of a component failure, another OR-gate is introduced and the tree assumes the form shown in Figure 10-4.

Here again the diamond indicates a set of conditions whose causes are chosen not to be developed further. Notice that the fault spelled out in the rectangle is a specific case of the top event with a more detailed description as to cause.

Now it might happen that the tank could miraculously withstand continuous pumping for $t > 60$ seconds but an application of the "No Miracles" rule constrains the situation such that the tank is always assumed to rupture under these conditions. This can be indicated on the fault tree by using an Inhibit gate whose input is "continuous pump operation for $t > 60$ seconds" (see Figure 10-5).

Can the input event to the Inhibit gate consist of a component failure? No, the pump is simply operating and pump operation for any length of time cannot consist of a component failure. Therefore this fault event must be classified "state of system." Now recall the rules of Chapter 4. Below a "state of system" fault there can be an OR-gate, an AND-gate, or no gate at all. Furthermore, the minimum, immediate, necessary and sufficient cause or causes should be sought. In this case the immediate cause is "motor runs for $t > 60$ seconds," a "state of system" fault. Its immediate cause is "power applied to motor for $t > 60$ seconds," a "state of system" fault. The immediate cause of the latter event is "K2 relay contacts remain closed for $t > 60$

seconds." The string of events shown in Figure 10-6 is added to the model to reflect this additional causal information.

In this case, nothing is lost by jumping from "pump operates continuously for t > 60 seconds" directly to "K2 relay contacts remain closed for t > 60 seconds." There is, however, no harm done in detailing the intermediate causes and, as a matter of fact, the opportunity for error is thereby lessened.

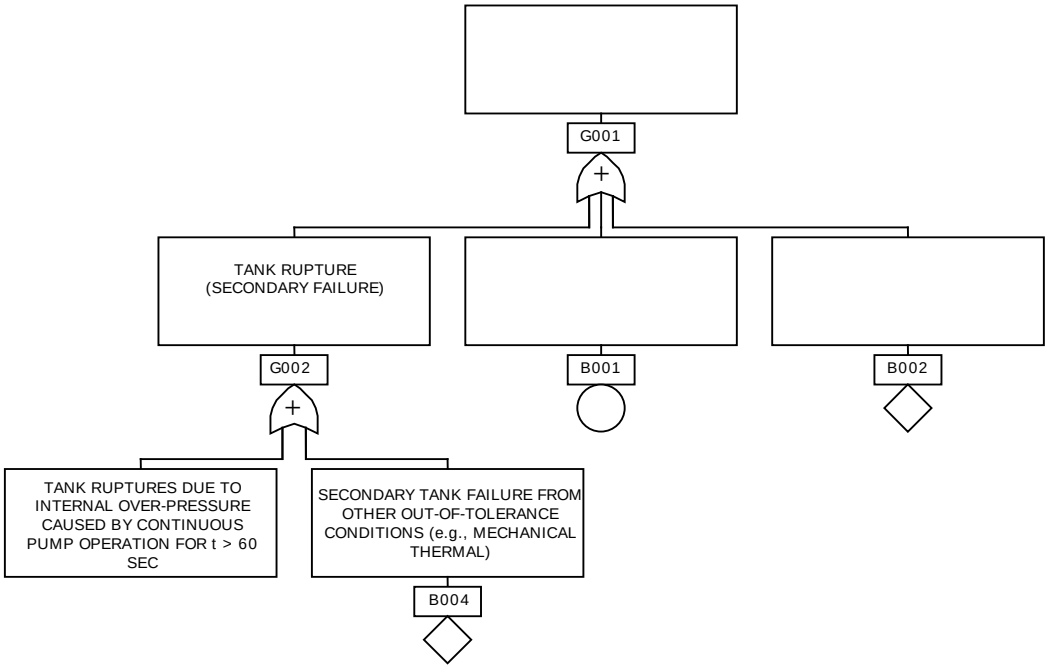


Figure 10-4. Fault Tree Construction – Step 2

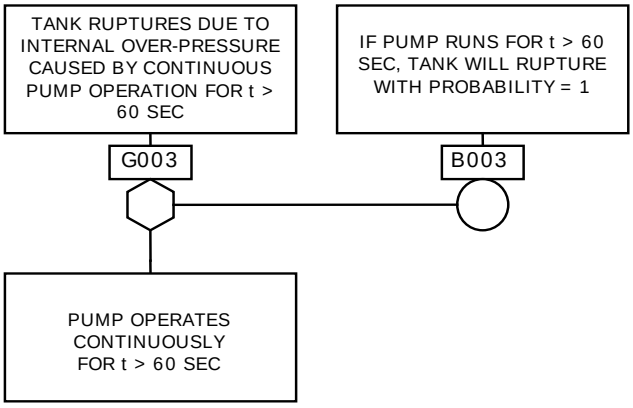


Figure 10-5. Fault Tree Construction – Step 3

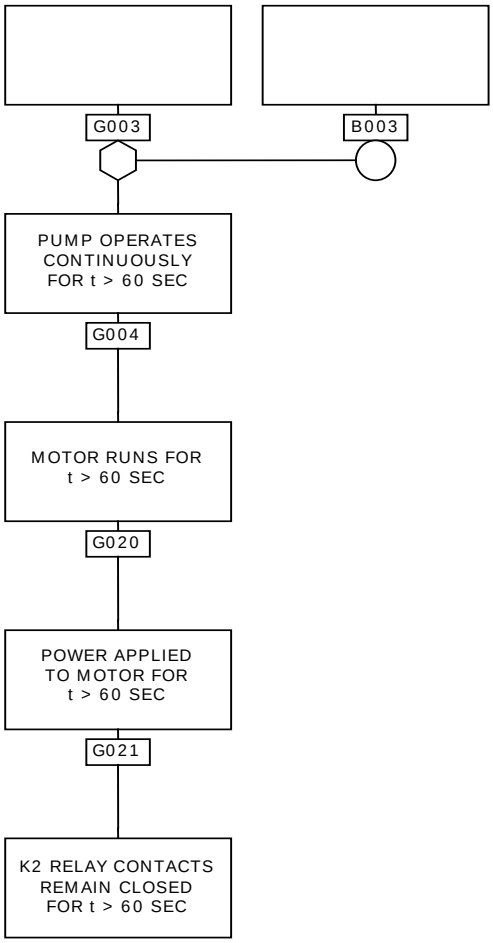


Figure 10-6. Fault Tree Construction – Step 4

Now consider the fault event, "K2 relay contacts closed for t > 60 seconds." Can this consist of a component failure? Yes, the contacts could jam, weld, or corrode shut. Therefore an OR-gate is drawn and primary, secondary, and command modes are added as shown in Figure 10-7.

The event of interest here is the command mode event described in the rectangle. Recall that a command fault involves the proper operation of a component, but in the wrong place or at the wrong time because of an erroneous command or signal from another component. In this case, the erroneous signal is the application of EMF to the relay coil for more than 60 seconds. This "state of system" fault can be analyzed as shown in Figure 10-8.

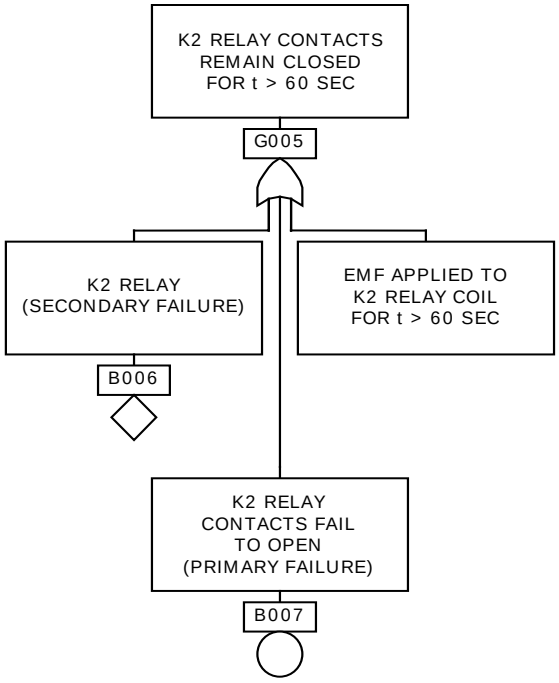


Figure 10-7. Fault Tree Construction – Step 5

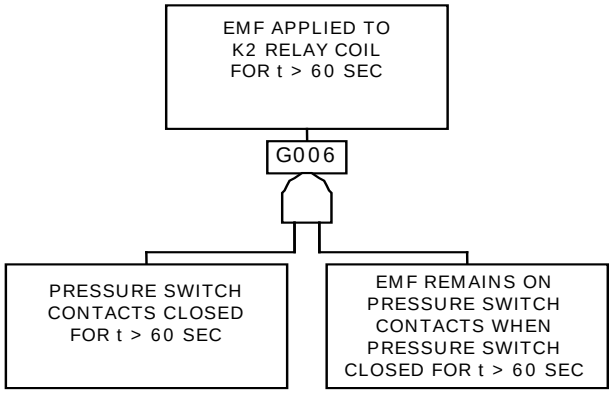


Figure 10-8. Fault Tree Construction – Step 6

Notice that both input events to the AND-gate in Figure 10-8 are written as faults. In fact, as has been discussed, all events that are linked together on a fault tree should be written as faults except, perhaps, those statements that are added simply as remarks (e.g., statements in ellipses). The pressure switch contacts being closed is not a fault per se, but when they are closed for greater than 60 seconds, that is a fault. Likewise the fact that an EMF is applied to the pressure switch contacts is not itself a fault. Notice that the condition that makes this event a fault is framed in terms of the other input event to the AND-gate.

The fault event, "pressure switch contacts closed for t > 60 seconds," can consist of a component failure, so both input events in Figure 10-8 are followed by OR-gates. These events are analyzed separately, starting with the left-hand event (see Figure 10-9).

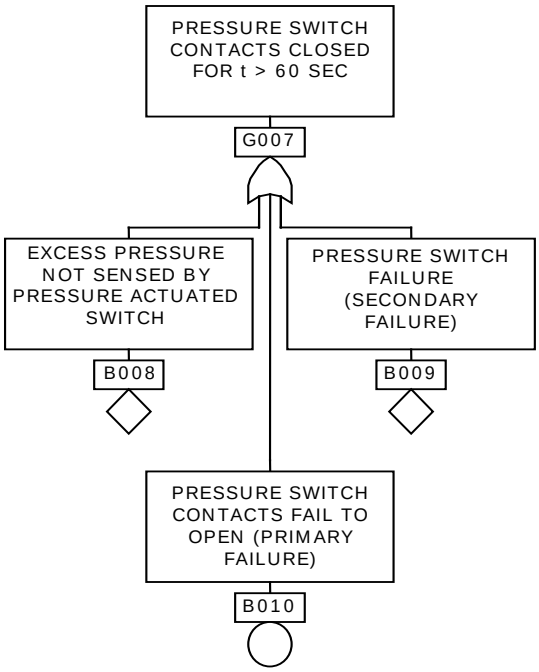


Figure 10-9. Fault Tree Construction – Step 7

This leg of the tree has reached its terminus (all input events are either circles or diamonds) unless, for some reason, it is desired to pursue the event in the left-hand diamond somewhat further (e.g., ruptured diaphragm, etc.).

The right-hand event in Figure 10-8 is now analyzed as shown in Figure 10-10.

Both the input events in Figure 10-10 are state-of-component faults. The left-hand one is the more easily analyzed as shown in Figure 10-11.

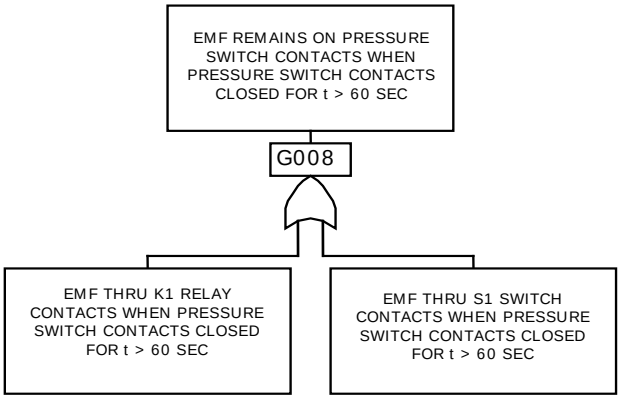


Figure 10-10. Fault Tree Construction – Step 8

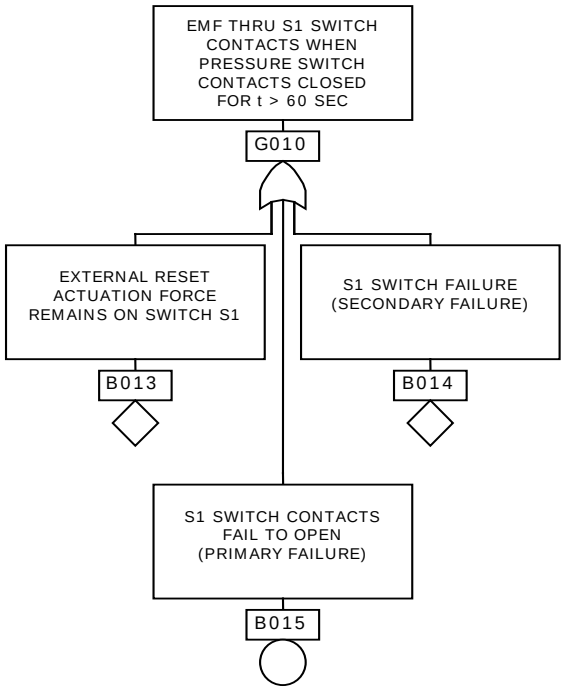


Figure 10-11. Fault Tree Construction – Step 9

Another tree terminus has now been reached. The analysis of the remaining input event in Figure 10-10 is shown in Figure 10-12. The reader will note from this latter figure that the fault tree development has continued—in this step-by-step fashion—down to timer relay faults. Finally, the complete fault tree for the pressure tank example is shown in Figure 10-13.

Actually, the fault tree of Figure 10-13 could be considered *too* complete. Because the only secondary failure developed was the rupture of the pressure tank due to overpumping, other secondary failures (the dotted diamonds) could simply be omitted from the diagram. Further simplifications can also be made, leading to the basic fault tree of Figure 10-14, where the basic events represent primary failures as shown in the legend and the fault events El, E2 etc. are defined as follows:

The E's are fault events.

- El - Pressure tank rupture (top event).
- E2 - Pressure tank rupture due to internal overpressure from pump operation for t > 60 seconds which is equivalent to K2 relay contacts closed for t > 60 seconds.
- E3 - EMF on K2 relay coil for t > 60 seconds.
- E4 - EMF remains on pressure switch contacts when pressure switch contacts have been closed for t > 60 seconds.
- E5 - EMF through KI relay contacts when pressure switch contacts have been closed for t > 60 seconds, which is equivalent to timer relay contacts failing to open when pressure switch contacts have been closed for t > 60 seconds.

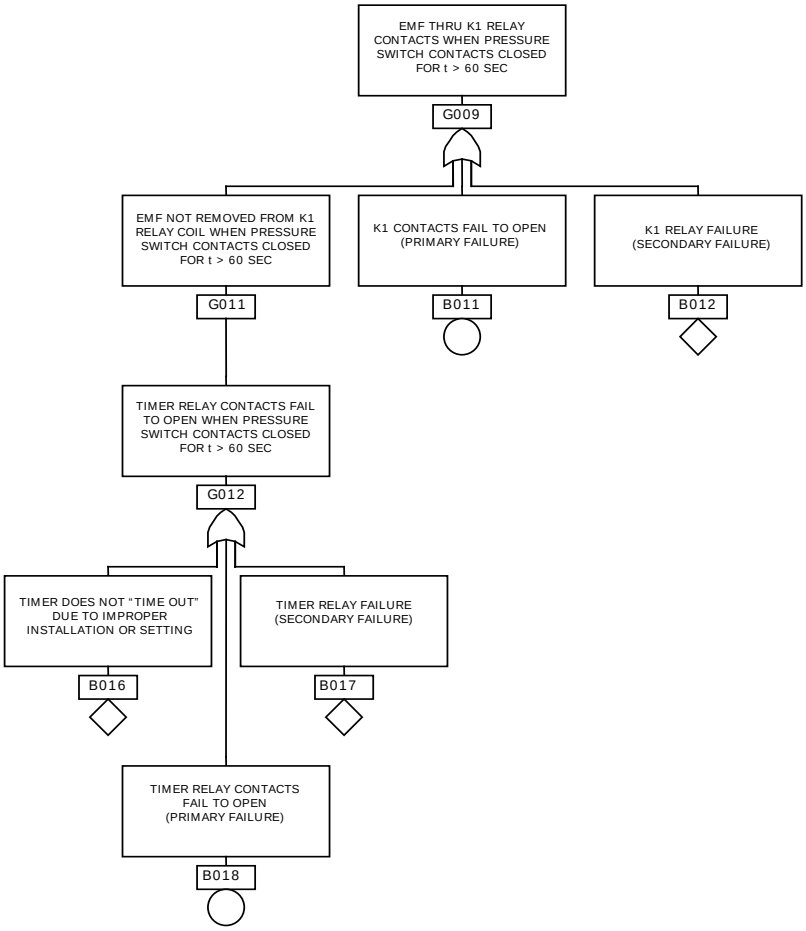


Figure 10-12. Fault Tree Construction – Final Step

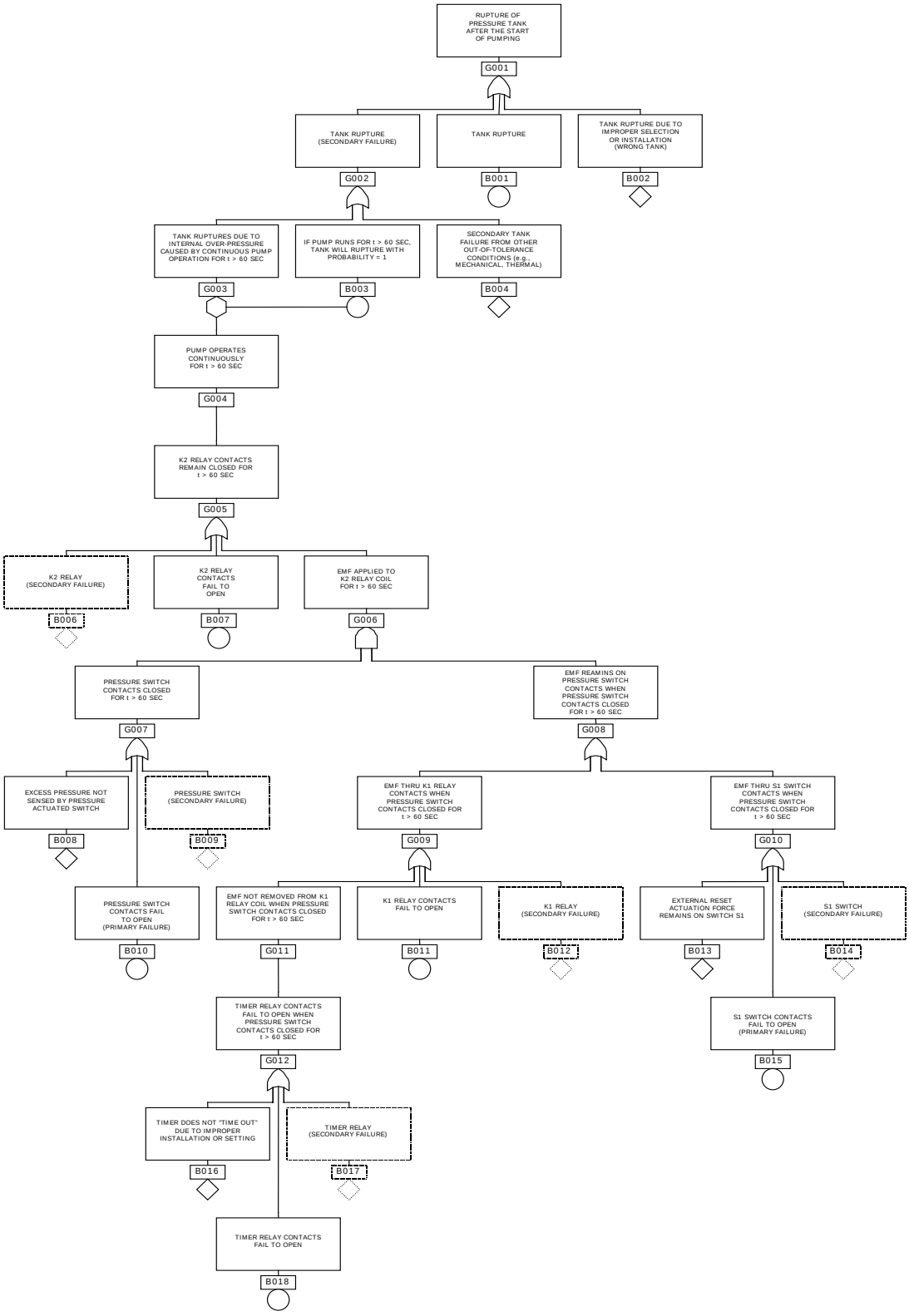
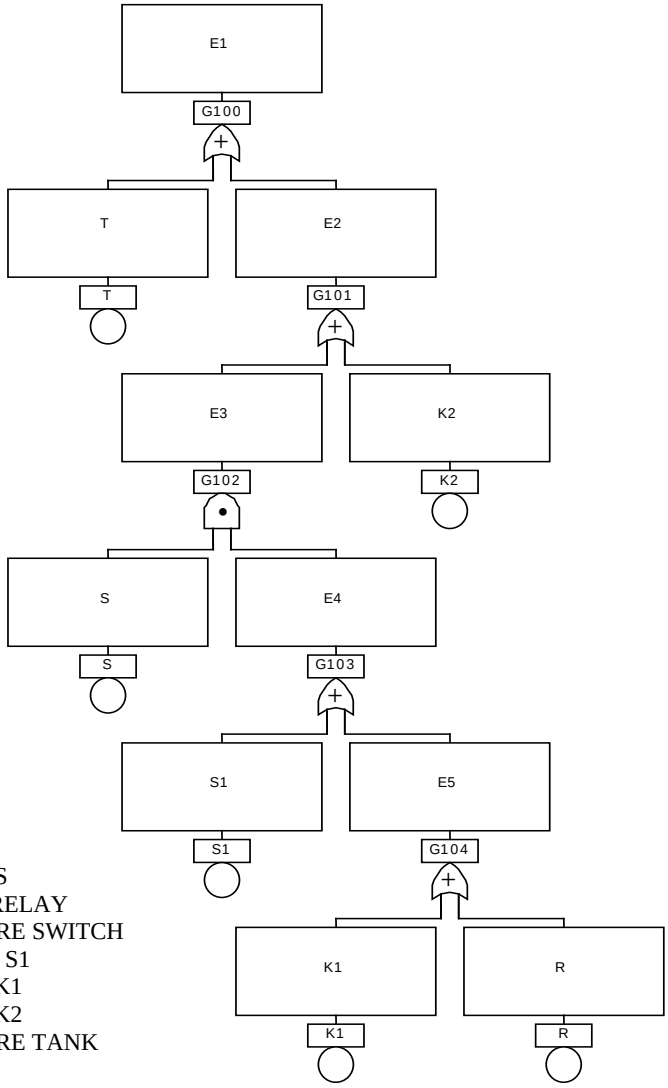


Figure 10-13. Pressure Tank Rupture Fault Tree Example



LEGEND: FAULTS

E1	TOP EVENT
E2, E3, E4, E5	INTERMEDIATE FAULT EVENTS
R	PRIMARY FAILURE OF TIMER RELAY
S	PRIMARY FAILURE OF PRESSURE SWITCH
S1	PRIMARY FAILURE OF SWITCH S1
K1	PRIMARY FAILURE OF RELAY K1
K2	PRIMARY FAILURE OF RELAY K2
T	PRIMARY FAILURE OF PRESSURE TANK

Figure 10-14. Basic (Reduced) Fault Tree for Pressure Tank Example

10.2. Fault Tree Evaluation (Minimal Cut Sets)

The top event of the fault tree shown in Figure 10-14 can be expressed as a Boolean function of the primary input events using the method explained in Appendix A, Section A.2. This is accomplished by starting at the top of the tree and working down:

El = T+E2
= T+(K2+E3)
= T+K2+(S•E4)
= T+K2+S•(S1+E5)
= T+K2+(S•S1)+(S•E5)
= T+K2+(S•S1)+S•(K1+R)
= T+K2+(S•S1)+(S•K1)+(S•R)

This expression of the top event in terms of' the basic inputs to the tree is the Boolean algebraic equivalent of the tree itself. E1 appears as the union of various combinations (intersections) of basic events and is the minimal cut expression for the top event. In the example, five minimal cut sets—two singles and three doubles—have been found:

K2
T
S•S1
S•K1
S•R

Each of these defines an event or series of events whose existence or joint existence will initiate the top event of the tree.

At this point, first, a qualitative assessment of results and then, armed with some data, a gross quantitative assessment can be made. Qualitatively, the leading contributor to the top event is the single relay K2 because it represents a primary failure of an active component. Therefore, the safety of the system would be considerably enhanced by substituting a pair of relays in parallel for the single relay K2. Actually, however, the system contains a much more serious design problems—the controls are being monitored instead of the parameter of interest (pressure in this case). It should be just the other way around! Thus, the most obvious way to improve the system would be to install a pressure relief valve on the tank and remove the timer.

The next basic event, in order of importance, is T, the primary failure of the pressure tank itself. Because the tank is a passive component, recall from Chapter 4 that the probability of the event T should be less (by an order of magnitude or so) than the probability of event K2. Of less importance are the three double component cut sets S•S1, S•K1, and S•R2, although it should be noted that the failure of the pressure switch contributes to each of them.

To make a quantitative assessment of the results requires estimates of failure probabilities for the components. Table 10-1 provides assumed values for the failure probabilities for the components in the system.

Table 10-1. Failure Probabilities for Pressure Tank Example

COMPONENT	SYMBOL	FAILURE PROBABILITY
Pressure Tank	T	5×10^{-6}
Relay K2	K2	3×10^{-5}
Pressure Switch	S	1×10^{-4}
Relay K1	K1	3×10^{-5}
Timer Relay	R	1×10^{-4}
Switch S1	S1	3×10^{-5}

Because a minimal cut is an intersection of events, the probabilities associated with the five minimal cut sets are obtained by multiplying the appropriate component failure probabilities (assuming independence of failures);

$$P[T] = 5\times10^{-6}$$

$$P[K2] = 3\times10^{-5}$$

$$P[S\bullet K1] = (1\times10^{-4})\,(3\times10^{-5}) = 3\times10^{-9}$$

$$P[S\bullet R] = (1\times10^{-4})\,(1\times10^{-4}) = 1\times10^{-8}$$

$$P[S\bullet S1] = (1\times10^{-4})\,(3\times10^{-5}) = 3\times10^{-9}$$

The probability of the top event, El is what is to be estimated. Observing that the top event probability is given by the probability of the union of the minimal cut sets, and that the probability of each individual minimal cut set is low, the rare event approximation (Chapter 7, Section 7.1) can be used. Therefore the minimal cut set probabilities are simply summed to produce

$$P(E_1) \cong 3.4\times10^{-5}.$$

The relative quantitative importance of the various cut sets can be obtained by taking the ratio of the minimal cut set probability to the total system probability:

Cut Set	Importance
T	14%
K2	86%
S•K1	Less than 0.1%
S•R	
S•S1	

The pressure tank example has been provided on numerous occasions as a workshop exercise for students learning the basics of fault tree construction. The most frequent analytical error made by these students is the tendency to leap directly from the event of rupture to the pressure switch. When this is done the single failure minimal cut set K2, i.e., the primary failure of K2, is missed completely. The tendency for this error to occur illustrates the importance of applying the rules described in Chapter 4.

One design-related principle that arises out of this example is that a system should be designed so that in the failure logic AND-gates appear as close to the tree-top as possible in an effort to eliminate single event cut sets. The family automobile is a good example of a system which is not of this type, and the reader can amuse himself by making a lengthy list of single events that will immobilize his or her car.

11. Monopropellant Propulsion System Example

In this chapter an example fault tree for a hypothetical propulsion system is presented. Following a description of the hypothetical propulsion system, the fault tree for the top event is developed in a step-by-step fashion with the help of the rules described in Chapter 4. The fault tree is then solved and limited conclusions are drawn.

11.1 Mono-propellant propulsion system

Consider a simple, pressure fed, monopropellant propulsion system for a small space flight vehicle shown in Figure 11-1. The purpose of the system is to provide thrust for the vehicle while in orbit. Additional support systems (such as thermal control) that may be required for such a system are ignored for this example.

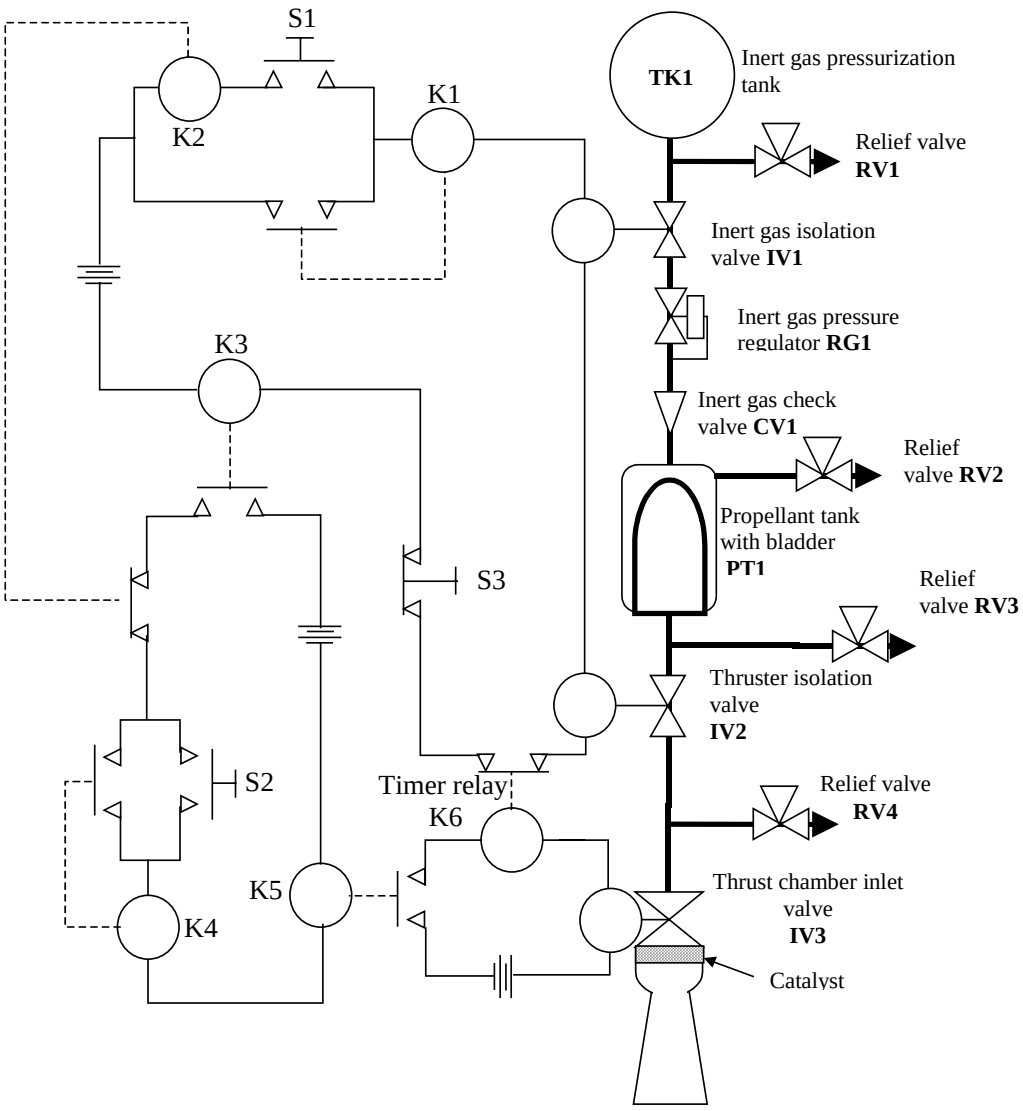


Figure 11-1. Monopropellant Propulsion System

The system uses hydrogen peroxide (H_2O_2) that passes over a catalyst and decomposes into byproducts of oxygen, water, and heat to create an expanding gas producing a thrust that changes the spacecraft velocity. The propellant system component nomenclature is listed in Table 11-1.

Table 11-1. Propellant System Nomenclature

TK1 – Propellant Storage Tank	PT1- Propellant Tank 1
RV1 – Relief Valve 1	K1 – Arming Relay K1
RV2 – Relief Valve 2	K2 – Firing Protection Relay
RV3 – Relief Valve 3	K3 – Arming Relay
RV4 – Relief Valve 4	K4 – Firing Relay
IV1 – Isolation Valve 1	K5 – Firing Relay
IV2 – Isolation Valve 2	K6 – Timing Relay
IV3 – Isolation Valve 3	S1 – Arming Switch
RG1 – Regulator 1	S2 – Firing Switch
CV1 – Check Valve 1	S3 – Emergency Cutoff Switch

The system consists of a reservoir TK1 of inert gas that is fed through an isolation valve IV1 to a pressure regulator RG1. The pressure regulator RG1 senses pressure downstream and opens or closes to control the pressure at a constant level. A check valve, CV1 allows passage of the inert gas to the Propellant Tank PT1. Separating the inert gas from the propellant is a bladder that collapses as propellant is depleted. Propellant is forced through a feed line to the Thruster Isolation Valve IV2 and then to the Thrust Chamber Inlet Valve IV3. For the Thruster to fire, the system must first be armed, by opening IV1 and IV2. After the system is armed, a command is sent to IV3, to open, allowing H_2O_2 into the thrust chamber. As the propellant passes over the catalyst, it decomposes producing the byproducts and heat and the expanding gas that creates the thrust. The relief valves RV1-4 are available to dump propellant overboard should an overpressure condition occur in any part of the system.

The electrical command system controls the arming and thrusting of the propellant system. To arm the system, switch S1 is momentarily depressed, allowing electromotive force (emf) to activate relay switches K1, K2 and K3, and open valves IV1 and IV2. K1 closes and sustains the emf through the arming circuit. K2 momentarily opens to preclude the inadvertent firing of the system during the transition to the armed mode, and closes when S1 is released. K3 closes in the firing circuit. The system is now armed with power supplied to sustain IV1 and IV2 in the open position.

When firing switch S2 is momentarily depressed, K4 closes sustaining the firing circuit. K5 closes completing the circuit for K6, which begins timing to a predetermined time for the thruster to fire. The completed circuit opens IV3 and thrusting begins. When K6 times out, it momentarily opens breaking the arming circuit and opening K1. Power is removed from the IV1 and IV2 relays and both valves are spring-loaded closed. K3 opens breaking the firing circuit,

which opens K4 and K5. IV3 is spring-loaded closed, and the system is in now in the dormant mode. Should K6 fail and remain closed after timing out, the system can be shut down manually by depressing S3, which breaks the arming circuit, opening K1 and closing IV1 and IV2. The firing circuit relay switch K3 will open breaking the firing circuit, which causes K4 and K5 to open. When K5 opens, IV3 will be spring-loaded closed, and the system will be in the dormant mode. System operational configurations are summarized in Figure 11-2.

11.2 Monopropellant Propulsion System Fault Tree Development

In this example there are three general system failure modes that can be assessed: (1) the failure of the system to provide propulsion when commanded; (2) the inadvertent firing of the system when it is not required; and (3) the continued system firing after the system has been commanded off. For this example failure mode (3) is addressed.

Since failure mode (3) is of concern, the first question to be asked is, “Can this thruster continue to thrust by itself?” The answer is no, because a thruster needs propellant to continue thrusting. Therefore this fault is a “state of system” fault. Because it is a “state of system” fault, either an AND or OR gate may be applicable. Immediate, necessary and sufficient causes of this failure mode are identified. Propellant only enters the thruster through IV3, but since IV3 cannot provide propellant by itself, IV2 is also of concern. Both IV3 and IV2 must remain open to provide propellant to the thruster from propellant tank PT1. The fault tree is begun as shown in Figure 11-3.

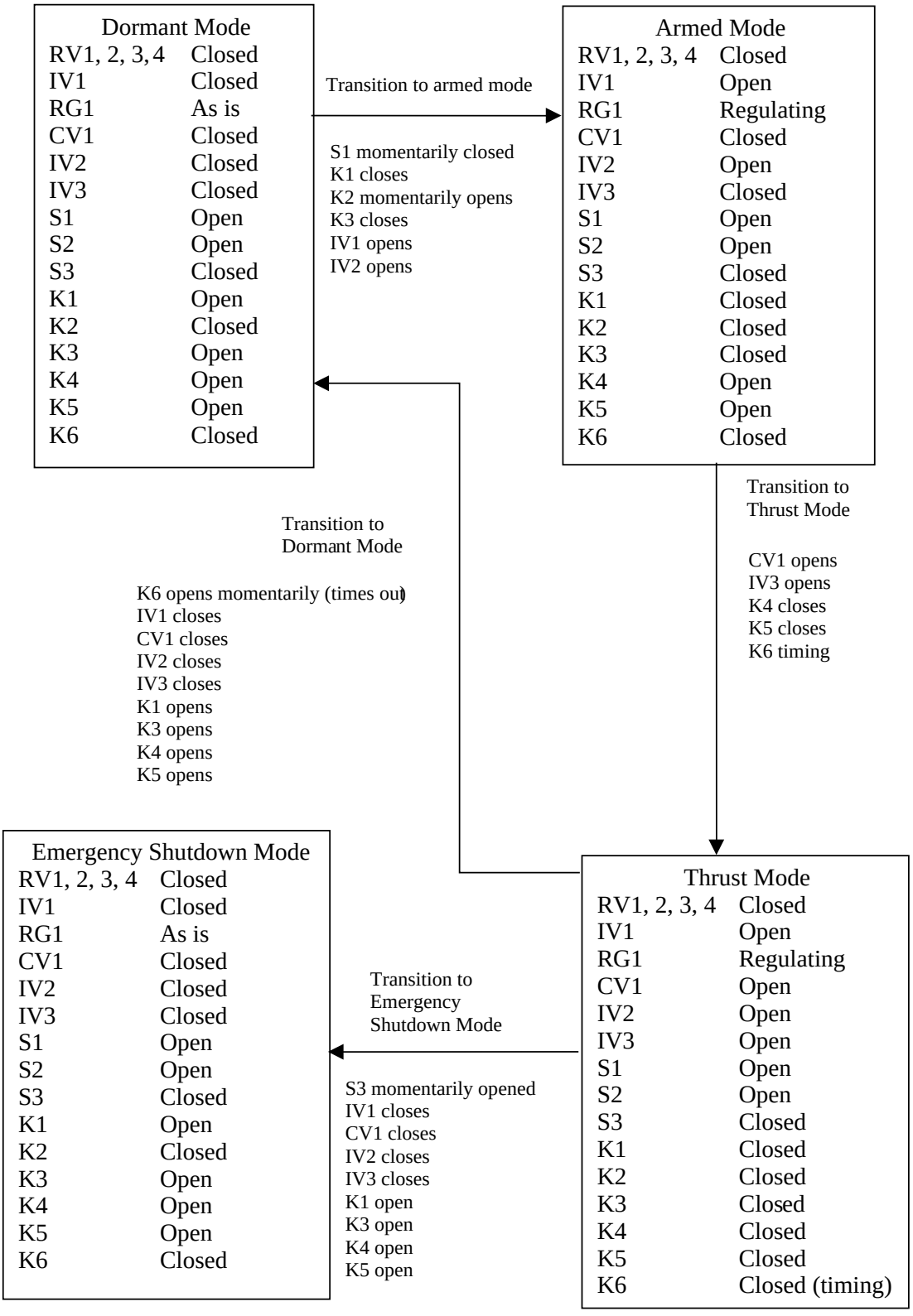


Figure 11-2. Monopropellant Propulsion System Operational Configurations

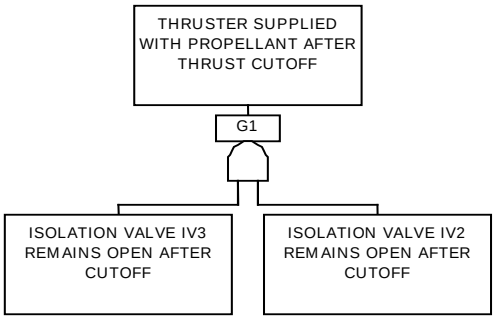


Figure 11-3. Fault Tree Construction – Step 1

It is assumed in this design that given the failure of IV2 and IV3 when commanded, and assuming that IV1 closes when commanded, there will be enough residual pressure in the propellant tank PT1, to blow down enough propellant to adversely affect the intended thrust.

Because they are spring-loaded, valves IV3 and IV2 are expected to close when the power is removed from their respective relays, the ways in which the valves can remain open are explored. A primary failure of IV3 (IV3 remains open even though emf is removed from its relay), can occur if a fault occurs in the structure of the armature; e.g., corrosion, contamination, or a structural failure may preclude the valve from closing. This is a primary failure of this component and is described as:

Primary failure of IV3 to close after cutoff.

This is a "state of component" failure because the fault can exist as a component failure. The gate is an OR gate, one side of which represents the component failure causes (e.g., corrosion, contamination, or structural failure). The other side of the OR gate consists of any other system-related causes that result in the same fault, that is, IV3 fails to close after cutoff. In this case, emf continues to be supplied to the IV3 relay. The fault tree continues, as shown in Figure 11-4. The fault also depends on the circuit that keeps IV3 open and can only occur with a fault in K5 that must remain closed to maintain current to the IV3 relay. This is a "state of component" fault that requires another OR gate with primary failure on one side and other sources of continued emf on the other. Fault tree construction continues in Figure 11-5.

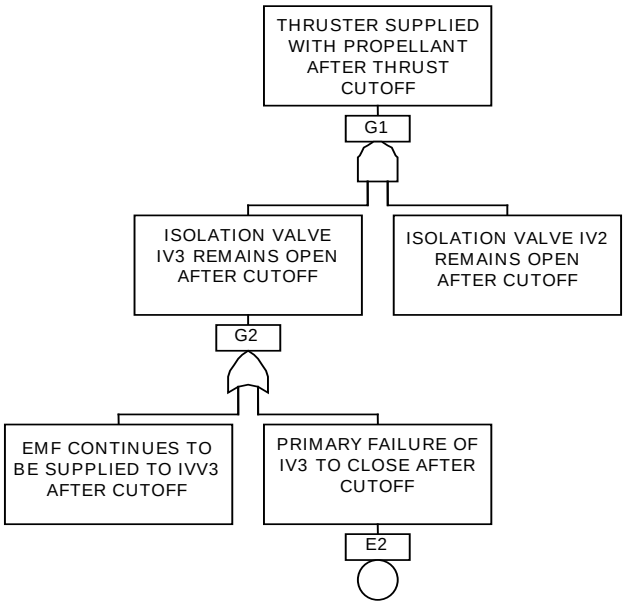
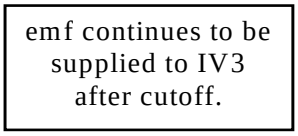


Figure 11-4. Fault Tree Construction – Step 2

In the same way the branch for Isolation Valve IV2 is constructed further to consider primary and command inputs. For the system side of the IV2 fault, it can be seen that one of two events must occur to remove emf from the IV2 relay; K6 must open at cutoff when the timer times out, or the emergency switch S3 must open when the operator observes that thrust is continuing and takes action. The S3 event is broken down into two events, an operation failure (sometimes called a command event) and a primary failure. In this case the assumptions is also made that an operator is observing the thrust time and will be ready to depress S3 should the K6 timer fail to automatically stop the thrust.



The fault, in Figure 11-5 is depends on the circuit that keeps IV3 open and can only occur with a fault in K5 (recall that “miracles” such as power supply failure or wire failures are not considered), which must remain closed to maintain current to the IV3 relay. This is a “state of component” and is another OR gate with primary failure on one side and other sources of emf on the other. The fault path leads to the following tree.

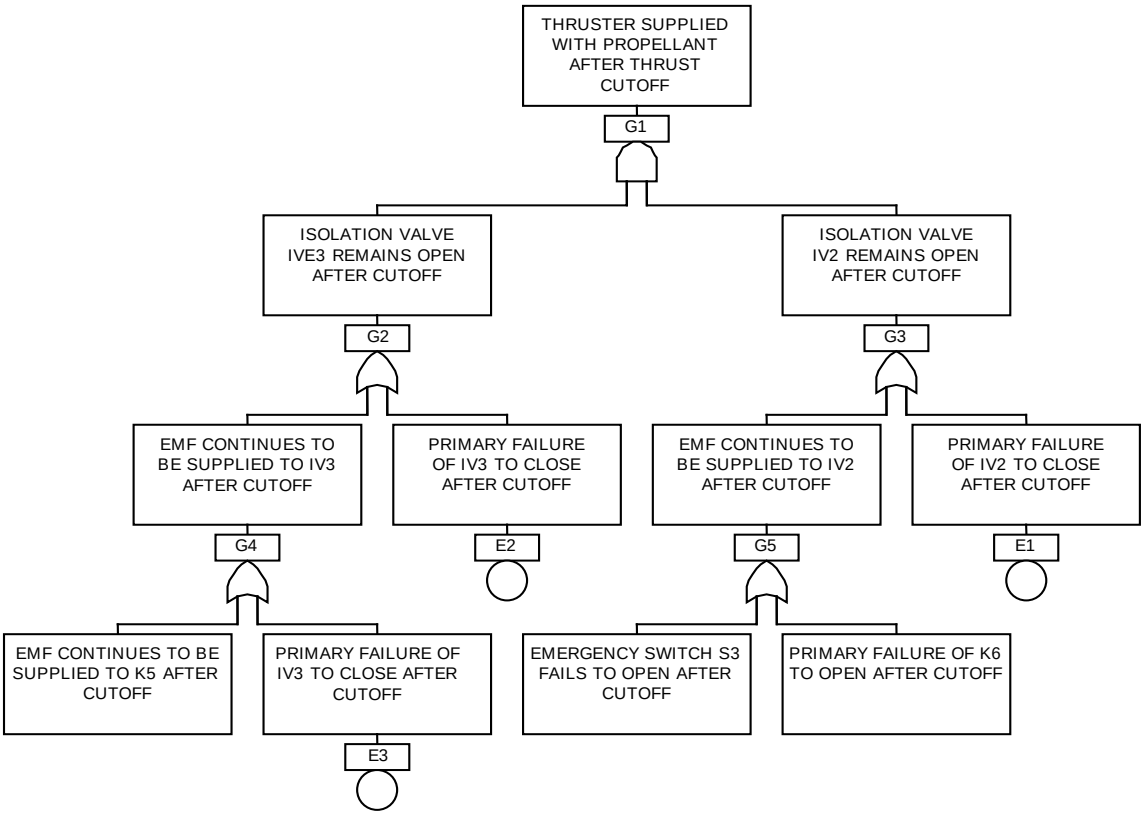


Figure 11-5. Fault Tree Construction – Step 3

On the IV3 side, the tree continues to be constructed through the firing circuit path back through relay K5 and then to relay K3, which will open when the K6 relay times out and momentarily opens. Relay K4 does not enter into the fault tree because its only purpose is to sustain the current in the firing circuit and can only open if relay K3 opens.

Note that the paths of each of the two major branches from the top event, that is IV3 and IV2, lead to a primary or operational failure of the emergency switch S3 and relay switch K6. A fault in either of these basic events will affect the propagation of events on both sides and could lead to the top event.

An additional point should be made regarding the "no miracles" rule. An event that is a "miracle" cannot occur to resolve a bad situation. For example, the battery deenergizing at the time that the thruster should stop is not considered.

Note that switch S1 can be depressed to stop the thrust, as well, since K2 will momentarily open. However, the arming circuit will still be activated (IV2 and IV3 open) and the intent of using S3 as an emergency stop is to place the system in the dormant mode.

The completed fault tree is shown in Figure 11-6. Note the transfers on the right side of the fault tree to gate G8 and G9 on the left side of the tree.

11.3 Qualitative and Quantitative Evaluation of the Fault Tree

Qualitative evaluation of the fault tree in Figure 11-6 is now performed to determine the minimum cut sets for continued thruster operation after cutoff.

The tree can be broken down into fault event gates as follows:

G1 – Thruster supplied with propellant after thrust cutoff
G2 – Isolation valve IV3 remains open after cutoff
G3 – Isolation valve IV2 remains open after cutoff
G4 – emf continues to be supplied to IV3 after cutoff
G5 – emf continues to be supplied to IV2 after cutoff
G6 – emf continues to be supplied to K5 after cutoff
G7 – emf continues to be supplied to K3 after cutoff
G8 – Emergency switch S3 fails to open after cutoff
G9 - Primary failure of K6 to open after cutoff

The basic events are listed below. The naming scheme used here is arbitrary, but the analyst should ensure that the naming scheme (see Section 5.6) specifically identifies the basic event failure mode. For ease in analysis, E1-E8 are used to keep the results simple

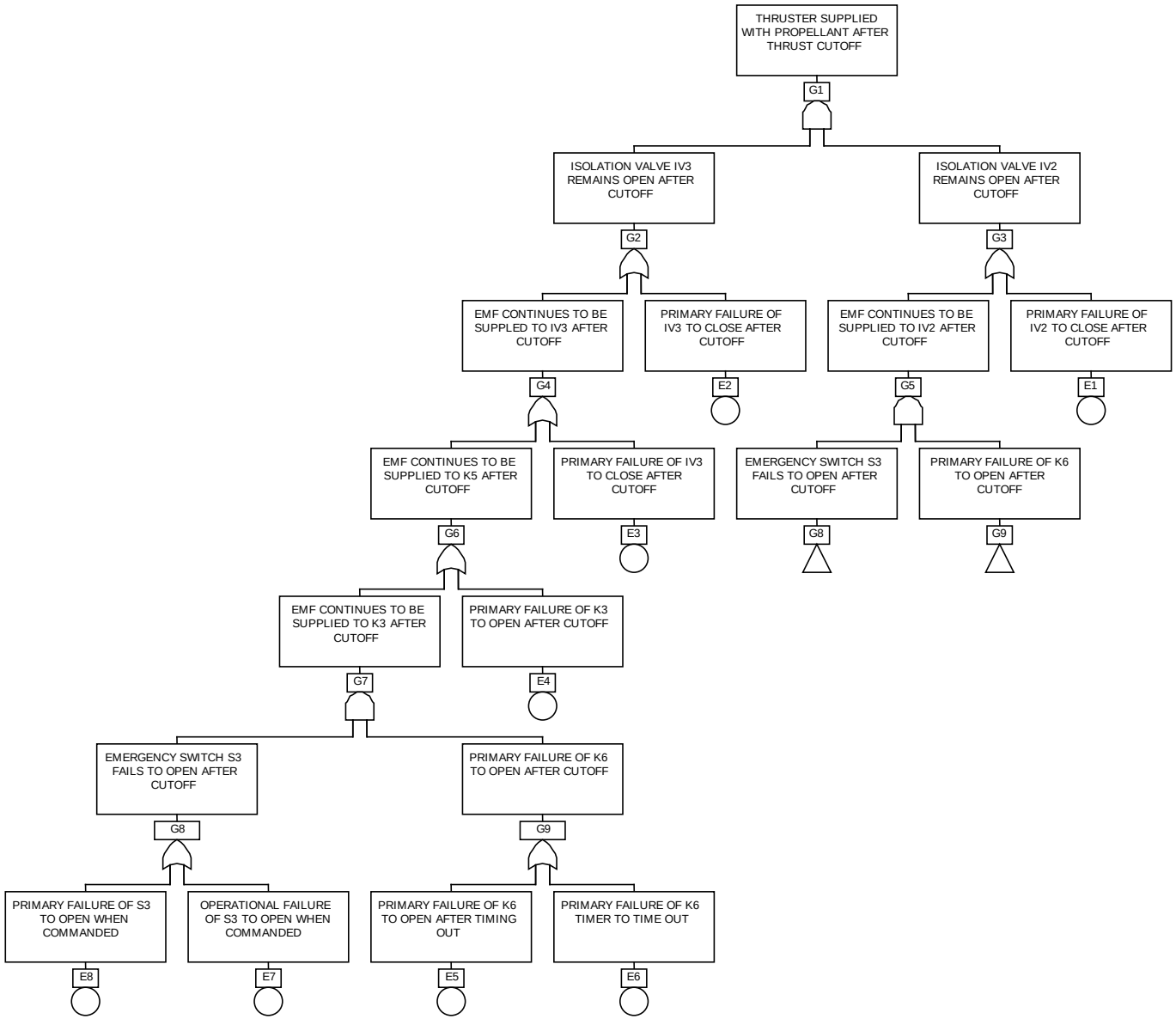


Figure 11-6. Thruster supplied with propellant after thrust cutoff.

E1 = Primary failure of IV2 to close after cutoff
E2 = Primary failure of IV3 to close after cutoff
E3 = Primary failure of K5 relay to open when emf is removed
E4 = Primary failure of K3 to open after cutoff
E5 = Primary failure of K6 to open after timing out
E6 = Primary failure of K6 timer to time out
E7 = Operational failure of S3 to open when commanded
E8 = Primary failure of S3 to open when commanded

In the equations below each fault event is expressed in terms of its equivalent Boolean equation. The gates are shown as G1-G9. The (•) indicates AND gates and plus sign (+) indicates OR gates.

G1 = G2 • G3
G2 = G4 + E2
G3 = G5 + E1
G4 = G6 + E3
G5 = G8 • G9
G6 = G7 + E4
G7 = G8 • G9
G8 = E7 + E8
G9 = E5 + E6

Next, starting from the bottom of the tree (Gate G9), each fault gate is expressed in terms of basic events.

G9 = E5 + E6
G8 = E7 + E8
G7 = (E7 + E8) • (E5 + E6)
G6 = ((E7 + E8) • (E5 + E6)) + E4
G5 = (E7 + E8) • (E5 + E6)
G4 = ((E7 + E8) • (E5 + E6)) + E4) + E3
G3 = ((E7 + E8) • (E5 + E6)) + E1
G2 = (((E7 + E8) • (E5 + E6)) + E4) + E3) + E2
G1 = ((((E7 + E8) • (E5 + E6)) + E4) + E3) + E2) • (((E7 + E8) • (E5 + E6)) + E1)

Expanding algebraically, the entire equation is:

(E7•E5)(E7•E5) + (E7•E5)(E7•E6) + (E7•E5)(E8•E5) + (E7•E5)(E8•E6) + (E7•E5)(E1) + (E7•E6)(E7•E5) + **(E7•E6)**(E7•E6) + (E7•E6)(E8•E5) + (E7•E6)(E8•E6) + (E7•E6)(E1) + (E8•E5)(E7•E5) + (E8•E5)(E7•E6) + **(E8•E5)**(E8•E5) + (E8•E5)(E8•E6) + (E8•E5)(E1) + (E8•E6)(E7•E5) + (E8•E6)(E7•E6) + (E8•E6)(E8•E5) + **(E8•E6)**(E8•E6) + (E8•E6)(E1) + (E4)(E7•E5) + (E4)(E7•E6) + (E4)(E8•E5) +(E4)(E8•E6) + **(E4)(E1)** + (E3)(E7•E5) + (E3)(E7•E6) + (E3)(E8•E5) + (E3)(E8•E6) + **(E3)(E1)** + (E4)(E7•E5) + (E4)(E7•E6) + (E4)(E8•E5) + (E4)(E8•E6) + (E4)(E1) + (E2)(E7•E5) + (E2)(E7•E6) + (E2)(E8•E5) + (E2)(E8•E6) + **(E2)(E1)**

Minimal cut sets are determined by reducing the expanded equation shown above. The first term in the equation is replicated cut set E7•E5. Since A•A=A (Appendix A), this term reduces to E7•E5. E7•E5 also appears as a part of many other terms in the equation. These terms are eliminated, since A+A•B=A. If the reader looks closely, there are only seven instances of unique combinations of basic events, which have been shown in bold text.

The equation reduces to the following minimal cut sets:

E6 • E7
E6 • E8
E5 • E7
E5 • E8
E1 • E3
E1 • E4
E1 • E2

Single point failures can be identified by the appearance of single basic events in the list of minimal cut sets. By simple inspection it can be seen that there are no single point failures.

Assuming the basic events have failure probabilities shown in Table 11-2, the minimal cut set probabilities are calculated to be as follows:

Table 11-2. Failure Probabilities for Monopropellant Propulsion System Example (the listed failure probabilities are hypothetical)

Basic Event	Component	Failure Mode	Failure Probability
IV	Isolation Valve	Failure to close when emf is removed	2×10^{-4}
K	Relay Switch	Failure to return when emf removed	3×10^{-3}
K6	Timer Relay Switch	Failure to time	2×10^{-2}
S	Manual Switch	Failure to close when commanded	1×10^{-5}
S	Manual Switch	Failure to open when commanded	5×10^{-5}
S	Operational Switch	Failure to command the switch open	1×10^{-2}

(E6 • E7) = 2.0×10^{-4}
(E5 • E7) = 3.0×10^{-5}
(E6 • E8) = 1.0×10^{-6}
(E1 • E3) = 6.0×10^{-7}
(E1 • E4) = 6.0×10^{-7}
(E5 • E8) = 1.5×10^{-7}
(E1 • E2) = 4.0×10^{-8}

Because the cut set probabilities are all small, the rare event approximation is applicable and the probability of the top event can be calculated by summing the cut set probabilities (2.3×10^{-4}).

One of the pitfalls in modeling is assuming that the probabilities of all failure modes of a particular component are the same. Depending on how the component is constructed, the probabilities will very likely be different for different failure modes, as in a valve failing open or close. In this example the calculated probability of continued system firing after the system has been commanded off is shown using hypothetical primary failure mode probabilities. The importance of using the proper failure mode probability could make a significant difference in the result.

11.4 Common Cause Failures

Note that the cut sets (E1 • E2), (E5 • E8), and (E1 • E4) result in probabilities that are very low, even for a redundant system.

The example shows several components that may be susceptible to common-cause failure (CCF), that is, situations in which the same type of components fail simultaneously for the same cause. For example, suppose all of the isolation valves had the same manufacturing flaw resulting in the valve failing to close when power was removed. The thruster could continue to burn despite the K6 relay timing out and the S3 emergency switch depressed simply because all three isolation valves remained open due to the same inherent fault.

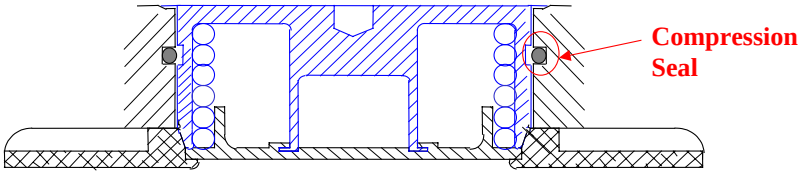
E1 and E2 (one of the cut sets), the primary failure of isolation valves IV2 and IV3 to close when commanded, are candidates for common cause failure investigation since the failure of both valves to close would result in the thruster continuing to fire (recall that the assumption has been made that enough pressurized propellant remains in the propellant tank PT1 to adversely affect the intended thrust). This can be accounted for in the fault tree by adding an extra basic event populated with the estimated CCF probability to gates G2 and G3.

12. Sample Container Seal Design Example

In this example the fault tree modeling process is applied to the design of a sealing mechanism for a container that is to return samples to Earth.

The first step in a risk-based design process is the establishment of a success criterion for the events of interest. In this case the success criterion is related to the effectiveness of the container in assuring containment of a sample. For the sake of this example assume a containment assurance requirement of 1.0×10^{-7}.

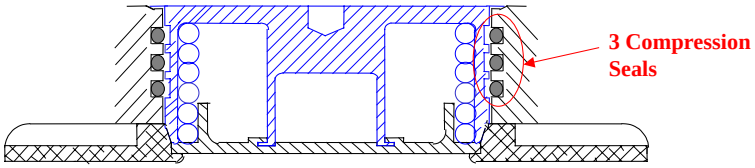
Figure 12-1 indicates a design of a simple single O-ring type seal of the type used in many applications, including the field joints of the shuttle solid rocket boosters*. To determine the risk involved in this simple design the probability of single O-ring failing to maintain containment must be established. This determination involves a thorough review of the historical performance of these types of seals made of the proposed material in the proposed application environment. From the investigation of this historical heritage a failure probability is chosen to represent the failure of this single seal under normal (that is, within specification) conditions. Now suppose this investigation discovers that these seals have been shown to be good with a level of assurance in the 3-sigma or 1.0×10^{-3} range (a fairly good assumption based upon the heritage of these designs). In this case the conventional design process would be unlikely to produce a design employing a single O-ring seal that would meet the success criterion. A simple conventional design solution, employing multiple seals is shown in Figure 12-2. Considering only independent failure the combined design would provide containment assurance in the 1.0×10^{-9} range, as shown by Fault Tree model given in Figure 12-3 (consistent with the rules for FT construction described in Chapter 4, the combined failure of the three seals is classified as a "state of system" fault and is modeled using an AND gate), which clearly meets the established success criterion.



- One seal on each side suffices to perform the sealing function

Figure 12-1. Conventional Functional Seal Design

* The shuttle boosters incorporate multiple O-rings in all joints.



- Three seals, rather than one, satisfies both functional and independent failure risk requirements
- Redundancy provides protection against independent seal failures

Figure 12-2. Risk-Based Design for Independent Failures

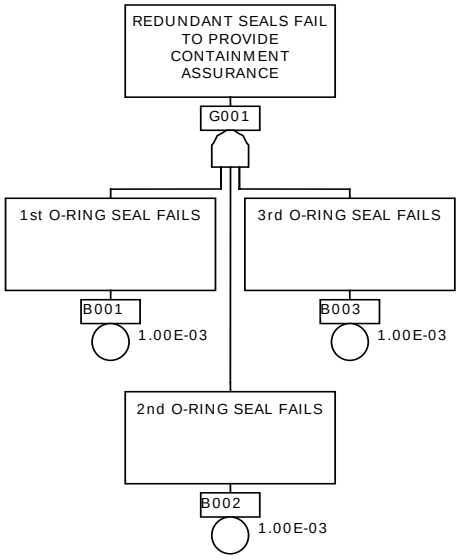


Figure 12-3. Fault Tree Model for Triply Redundant Design (Independent Failures Only)

However, in the case of the relatively simple example design given here, all failure modes are of concern for each seal, so the performance heritage for common cause failure modes must also be investigated as potential causes of loss of containment assurance. The full set of failure modes includes those that might represent a CCF to a triplicated identical design. Consideration of the heritage of non-diverse redundant sealing systems bears out their vulnerability to single causes that fail multiple seals (again the Shuttle SRBs are an example).

Modeling CCF potential using the conventional "Beta factor" method (Chapter 5, Section 5.2) and assuming a β-factor of 0.1 (i.e., 10% of the potential failures would represent a common cause threat) results in a CCF failure probability of $0.1 \times 1.0 \times 10^{-3}$, or 1.0×10^{-4}. Adding this failure to the fault tree results in a containment assurance failure probability of 1.0×10^{-4}, as shown in the

Fault Tree model given in Figure 12-4*. Consideration of CCF therefore places the design unacceptably above the level required by the success criterion.

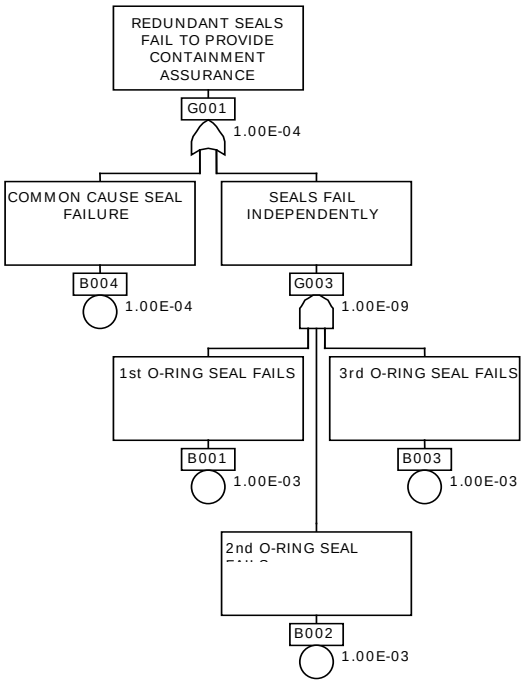


Figure 12-4. Fault Tree Model for Triply Redundant Design Considering Common Cause

To avoid this, the designer must consider what types of additional diverse design features might protect against or mitigate the potential for common cause failure. Figure 12-5 shows such a design. Here a metal-to-metal seal separates two O-ring seals and the O-rings are placed in different geometric planes. The use of a diverse metal-to-metal seal protects against common cause failure modes such as low temperature, which might impact the resiliency of the O-rings. The geometrical separation of the O-rings and their non-coplanar placement mitigates against the potential for a contaminating fiber or other extended debris compromising both O-rings simultaneously. Additionally, a fused plug seal has been added as a mitigator against more insidious debris environments such as dust or other finely suspended particles. The addition of these diverse elements in the design improves the common cause protection significantly. This improvement has been accounted for in a revised fault tree by reducing the CCF β-factor by an order of magnitude from 0.1 to 0.01. This improves the overall design performance to 1.0×10^{-5} as shown in the Fault Tree model given in Figure 12-6. This value reflects the fact that the seal insertion process is still vulnerable to dust contamination up to the last seal, the fuse plug, and so the final seal represents a single seal for residual contaminants given the plug insertion process assumed.

* Since the common cause seal failure fails the entire seal assembly, it can be considered to be a "state of component" fault. The fault tree has been revised to address this through the use of an OR gate for the top event.

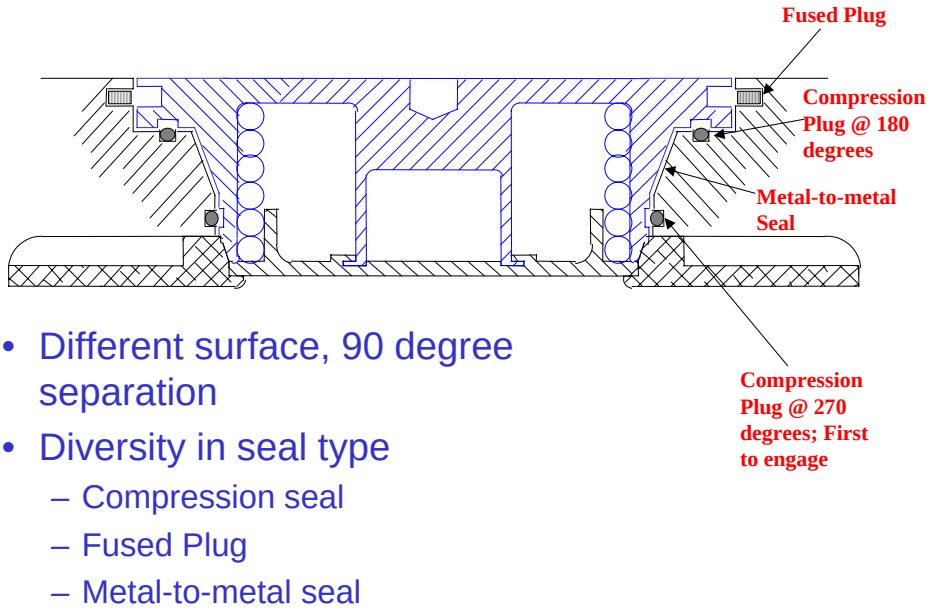


Figure 12-5. Risk-Based Design against CCF

The design effort then moves to improving the sealing sequence and identifying methods which, at most, would expose the first sealing surface to contamination.

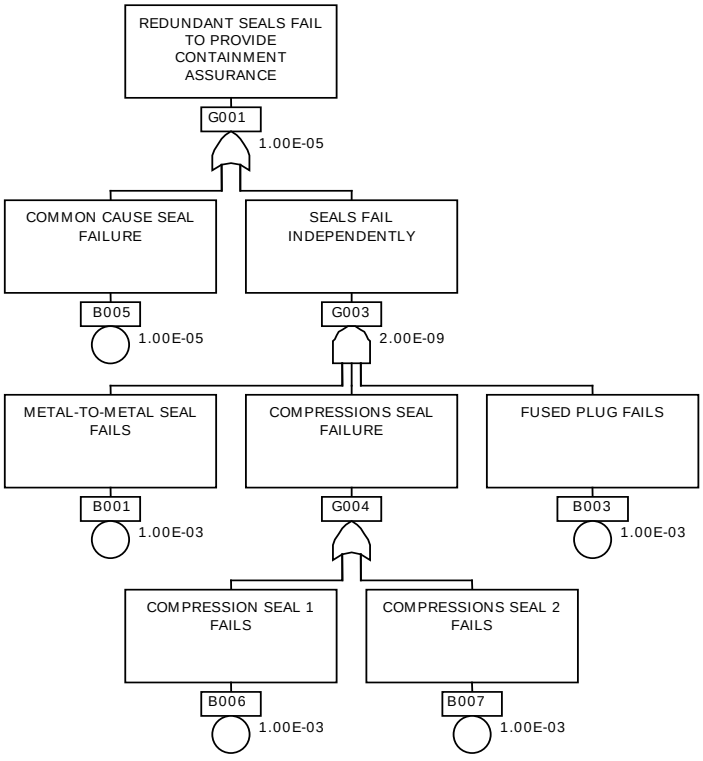


Figure 12-6. Fault Tree Model of Risk-Based Design Against CCF

Assume next that it is determined that a debris avoidance or removal process using a special tape would reduce further the vulnerability to common cause associated with widespread debris. The Figure 12-6 model is revised to require an additional failure (contamination tape fails) before an overall common cause seal failure would occur (Figure 12-7). This failure plus the previously described CCF basic event are included under an AND gate (since neither of these failures individually results in a common cause seal failure, this is a "state of system" fault). When the revised model is quantified, the overall probability improves to 1.0×10^{-6}, within striking distance of the requirement for design success.

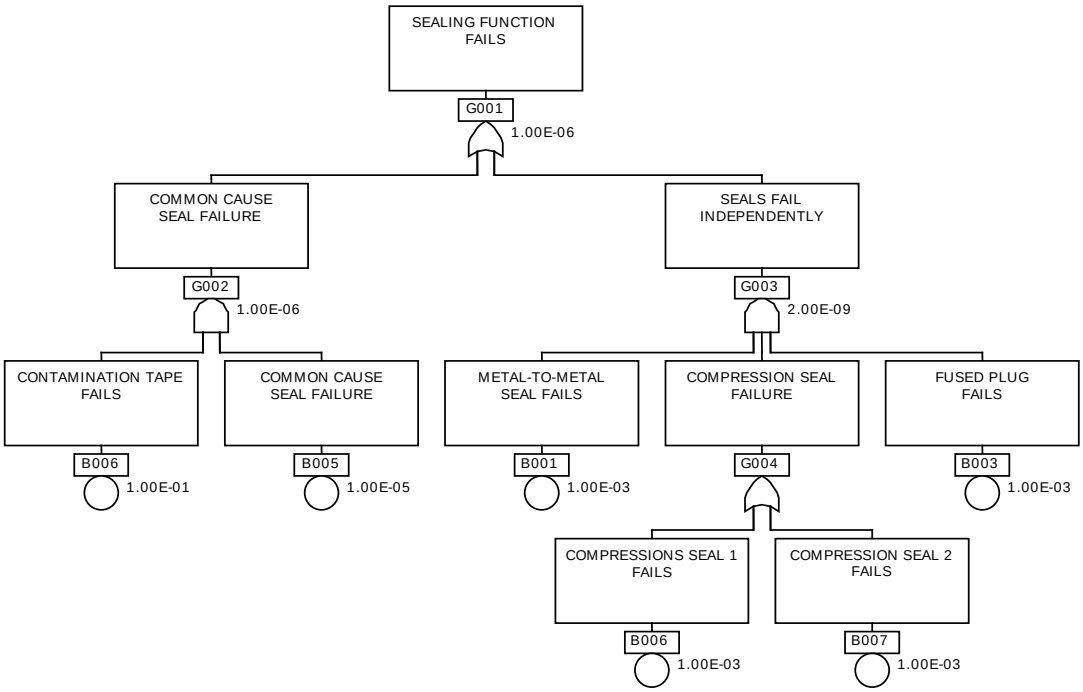


Figure 12-7. Fault Tree for Contamination and CCF Risk-Based Design

The design team would then focus attention toward adding more mitigating features to attack the residual risk by isolating the first seal from the environment as well and/or reducing the probabilities of failure of the individual seals by the choice or the development of different, more robust, sealing materials.

13. Hypothetical Computer System Example

This example demonstrates the use of the Dynamic Fault Tree constructs described in Chapter 8 for reliability analysis of fault tolerant computer based systems. Chapter 8 describes the specialty gates and specialty software developed specifically for modeling DFTs. Figure 1 shows a block diagram of a hypothetical example system that will be referred to as HECS. A description of the processing, memory and bus systems and applications that constitute HECS are described below, along with the modeling approach used.

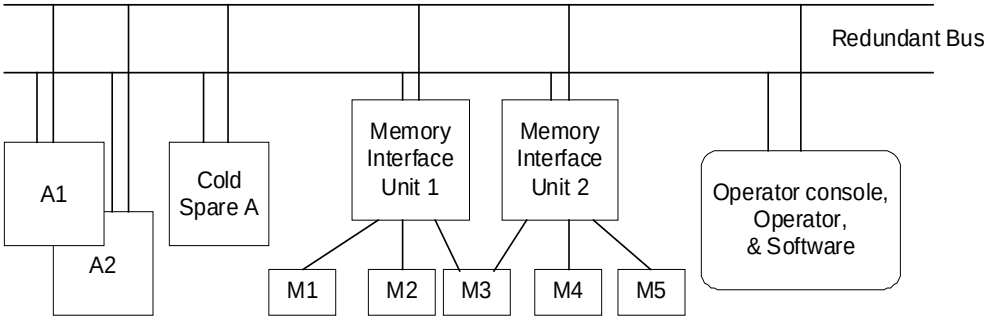


Figure 13-1. HECS (Hypothetical Example Computer System)

13.1 Developing the Fault Tree for HECS

The HECS system requires the correct operation of the processing, memory and bus subsystems, as well as the software application. Thus HECS will fail ("state of system" fault) if any of these subsystems fail. The top level of the HECS fault tree appears in Figure 13-2, where the four major subsystems are listed. The development of each of these subsystems will be investigated in turn.

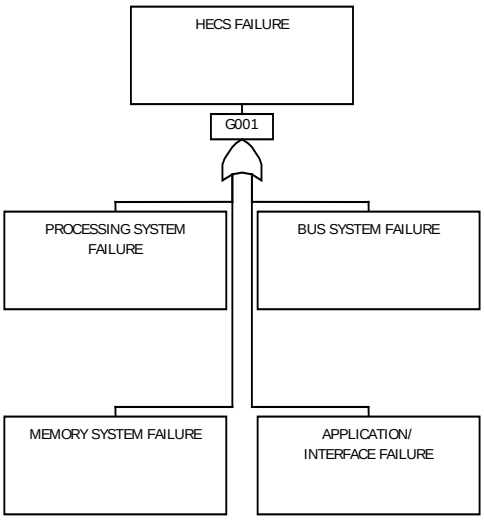


Figure 13-2. HECS Failure Causes

Modeling the processing system

HECS includes dual-redundant processors A1 and A2 and a cold spare, which can replace either upon failure. A cold spare is one that is assumed not to fail before being used. Processors A1, A2 and the cold spare A are all identical processors, running the same operating system. The processors are dual-redundant and use comparison monitoring for fault tolerance. Periodic checkpoints are taken and stored to aid in recovery from errors. Comparison of results and checkpoints are used to detect errors. When one of the two active processors is determined to have failed, the cold spare takes its place. The system can continue to operate until all three processors have failed.

Figure 13-3 shows the use of an AND gate to model the "state of system" processing system failure, as well as the use of the SPARE gate available in the Galileo/ASSAP software (see Chapter 1, Section 1.7), to model the relationship between the two active (hot) processors and the cold spare. The cold spare A is shared between the two active processors, thus its representative basic event inputs to both SPARE gates.

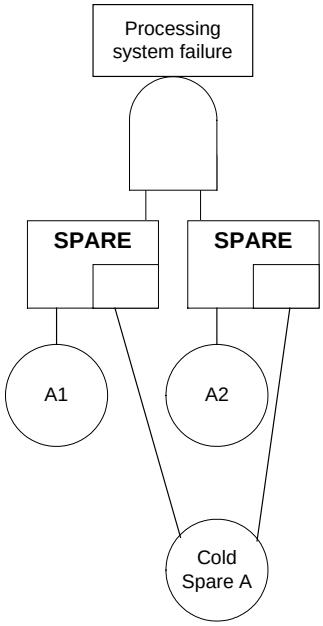


Figure 13-3. Fault Tree Model for HECS Processors

Modeling the memory system

HECS includes five memory units of which three are required. These memory units are connected to the redundant bus via two memory interface units. If a memory interface unit fails, the memory units connected to it are unusable. Memory unit 3 (M3) is connected to both interfaces for redundancy; thus M3 is accessible as long as either interface unit is operational.

The function of the memory units illustrates functional dependencies between elements. There are five memory units, of which three are required for reliable operation. Thus nominally, one would expect to connect the basic events for the five memory units using logic such that when

three of the five units fail, the memory system becomes unreliable. However, there is an added complication to consider. The memory units are connected to the busses via a pair of memory interface units. The memory interface unit must be operational in order for the memory unit to be accessible; thus the memory units are functionally dependent on the interface.

Figure 13-4 shows the portion of the fault tree that models the memory units. The five memory units (M1 through M5) are connected to a three-of-five combination gate, which provides failure logic such that the gate output is failed if three of the five inputs are failed (most fault tree software provides a combination gate as a modeling convenience). The dependency of the memory units on the interface is captured in the three functional dependency (FDEP) gates. The memory interface units are the trigger input to the FDEP gates, and the dependent basic events are the memory units. The FDEP gate operates by labeling the dependent basic events as failed when the trigger event occurs. The trigger event for an FDEP gate need not be a basic event; the second functional dependency gate triggers the failure of memory unit M3 when both interface units fail.

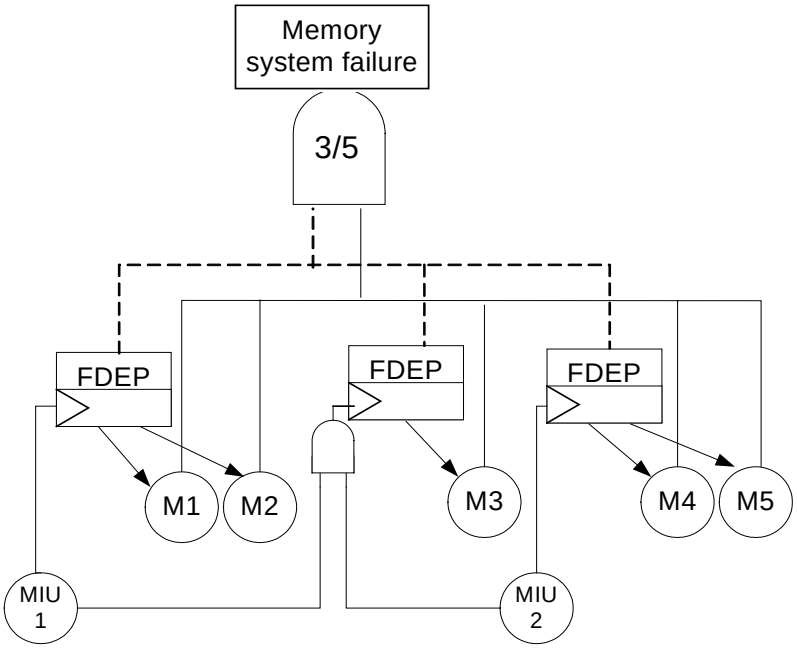


Figure 13-4. Fault tree for HECS Memory System

Since the FDEP gate does not produce a logical output that affects the fault tree output, it is connected to the fault tree via a dummy output signaled by a dashed line in the figure. Thus the three-of-five combination gate has five logical inputs, and three dummy inputs (the three-of-five specification relates to the number of logical inputs that have to fail, and not to the dummy inputs).

Modeling the bus system

The bus subsystem is a simple one to elaborate. There are two identical redundant buses, of which one is required for system operation. Thus the bus system fails when both of the buses

fail. Figure 13-5 shows the Galileo/ASSAP fault tree elaboration of the bus system failure. The basic event is labeled 2*Bus in order to represent the fact that the two buses are statistically and functionally identical.

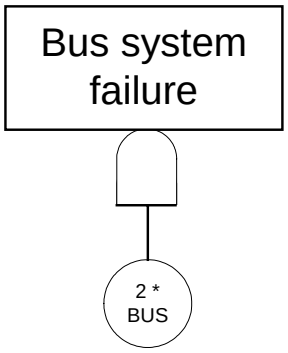


Figure 13-5. Fault Tree for HECS Bus System

Modeling the application failure

The last subsystem to be considered is the application subsystem. The application software runs on the computer system. The operator is a human who interfaces with the computer system via a Graphical User Interface (GUI) that runs on an interface device. Thus an application (software (SW)) failure, GUI (hardware (HW)) failure or human operator error will lead to system failure. Figure 13-6 shows the Galileo/ASSAP fault tree elaboration of the application and interface.

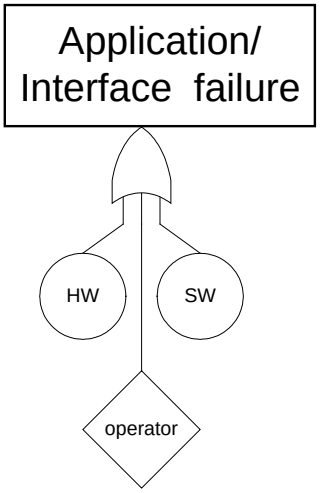


Figure 13-6. Fault Tree for HECS Application

HECS system level fault tree

The full HECS fault tree drawn by Galileo/ASSAP is shown in Figure 13-7.

13.2 Fault Tree Quantification

There are two kinds of failure parameters needed to quantify the basic events for the HECS system: failure rates or probabilities and coverage* parameters. Consider each subsystem in turn.

Failure rates and probabilities

Processing system

The failure rate for each the processors (when active) is assumed to be a fairly typical value for such systems, $\lambda_P = 10^{-4}$ per hour. Because the spare is cold, and thus is assumed not to fail before it is used to replace a failed processor, the dormancy factor* is zero.

Memory system

The failure rate for each of the memory units is assumed to be $\lambda_M = 6\times10^{-5}$ per hour, while that of the memory interface units is assumed to be $\lambda_{MIU} = 5\times10^{-5}$ per hour.

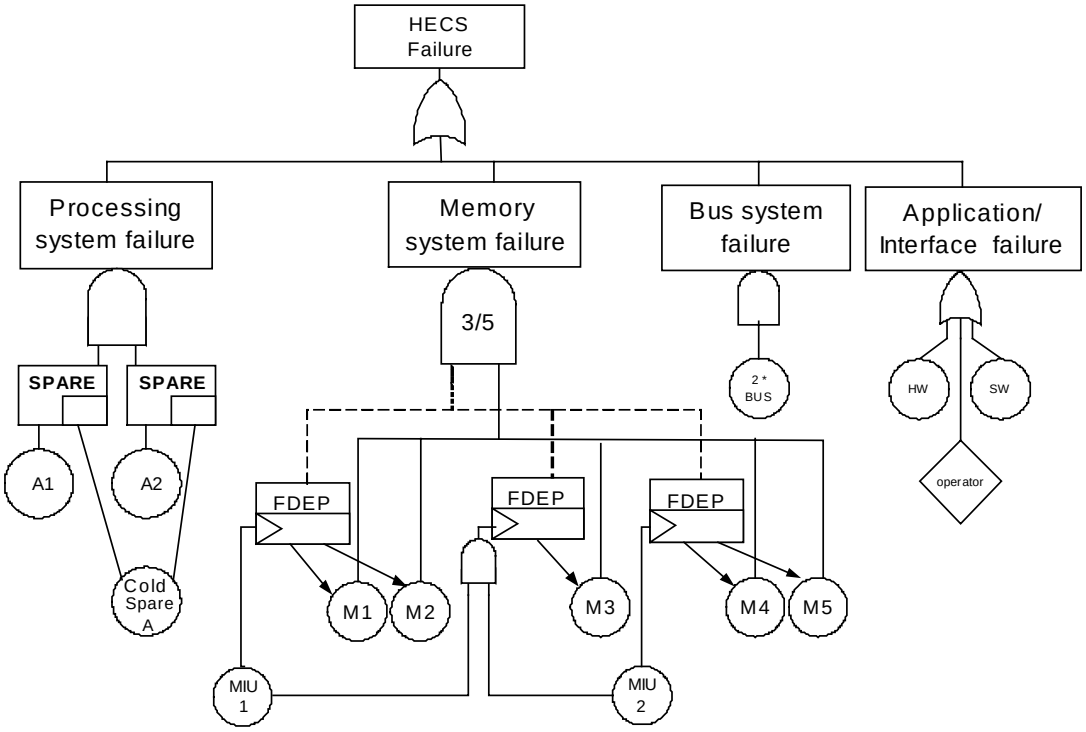


Figure 13-7. Fault Tree for HECS

* See Chapter 8 for a definition of this term.

Bus system

The failure rate for the each individual bus is assumed to be $\lambda_B = 10^{-6}$ per hour.

Application/Interface system

The failure rate for the GUI HW is assumed to be $\lambda_{HW} = 5\times10^{-5}$ per hour. The two remaining basic events are quantified in terms of failure probabilities rather than rates. The assumed probability of failure for the operator is $P_O = 0.001$, which means that, on average, the operator is assumed to be 99.9% reliable. A similar quantification for the application software results in a software reliability of 97%. Thus the failure probability for the application software is assumed to be $P_{SW} = 0.03$.

Coverage parameters

For the fault tolerant components, coverage parameters must be defined. Chapter 8 described the three coverage probabilities (*r*, *c* and *s*) representing, respectively, the probability of successful recovery from a transient fault, the probability of successful recovery from a permanent fault and the probability that a fault is uncovered. To estimate these parameters, some analysis of the fault tolerance mechanisms is in order.

Processing system

A processor contains built-in test functionality so that error checking occurs concurrently with instruction execution. If an error is detected, the instruction is retried immediately. Partial results are stored in case the retry is unsuccessful, so that the computation can be continued from some intermediate point (called a checkpoint). The process of continuing a computation from a previously saved checkpoint is called a rollback. In some cases the fault is such that the rollback is not successful, so the computation must start over after a system-level recovery procedure is invoked.

An example of a processor fault coverage model is shown in Figure 13-8, and represents the following hypothetical recovery procedure. First, assume that the fault is transient, and begin a four-step recovery procedure that continues as long as an error is detected. If an error persists after all steps have been performed, then a permanent recovery procedure must be invoked.

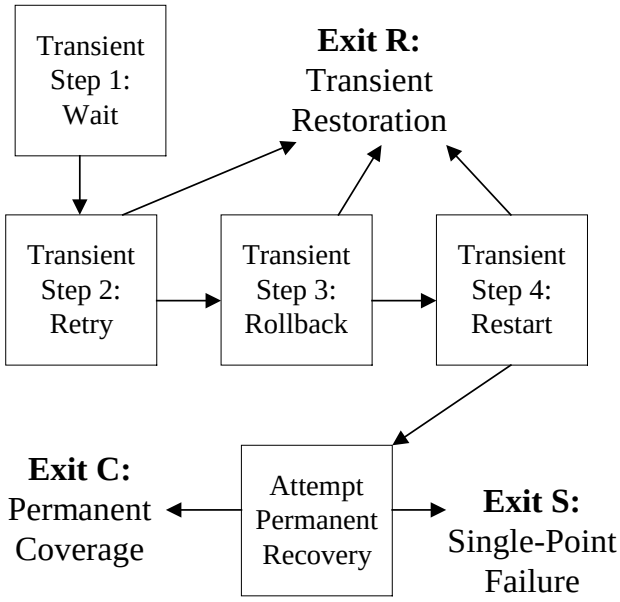


Figure 13-8. Coverage Model for HECS Processors

Step 1. Wait for a short time (a few cycles) and do nothing. If the fault is transient, it may disappear during this time, allowing rollback to succeed.
Step 2. Retry the current instruction several times.
Step 3. If an error persists, perform a rollback to a previous checkpoint, and pick up the computation from the checkpoint.
Step 4. Restart the processor and either reload a checkpoint or start the task from the beginning.

If an error persists after the four-step transient recovery process, it is assumed to be caused by a permanent fault. A system level permanent fault recovery process is begun, to remove the offending processor from the set of active units and to reconfigure the system to continue without it.

The analysis of this coverage model consists of calculating the probability of system recovery for each step of transient recovery and for permanent recovery, given parameters that define the probability of success and duration of each phase and the characteristics of faults themselves. The detail of the analysis is beyond the scope of this handbook. The results of the analysis of the coverage model are summarized in the three coverage probabilities, one for each of the three exits of the coverage model (Exits R, C and S). For this example, transient restoration is assumed to be an effective recovery procedure for 70% of all faults (r_p = 0.7). Permanent coverage is assumed to be 98% effective on the remaining 30% of faults (i.e., permanent faults). Thus, c_P = 0.294. The probability that a single processor fault is uncovered, and thus leads to system failure is $s_P = (1 - c_p - r_p) = 0.006$.

Memory system

A hypothetical recovery procedure for the memory units is shown in Figure 13-9. The memory uses an error correcting code, so a single-bit error is always detectable and correctable, and no

reconfiguration is required. If 98% of all memory faults affect only a single bit, then the probability of reaching the **R** exit is $r_M = 0.98$.

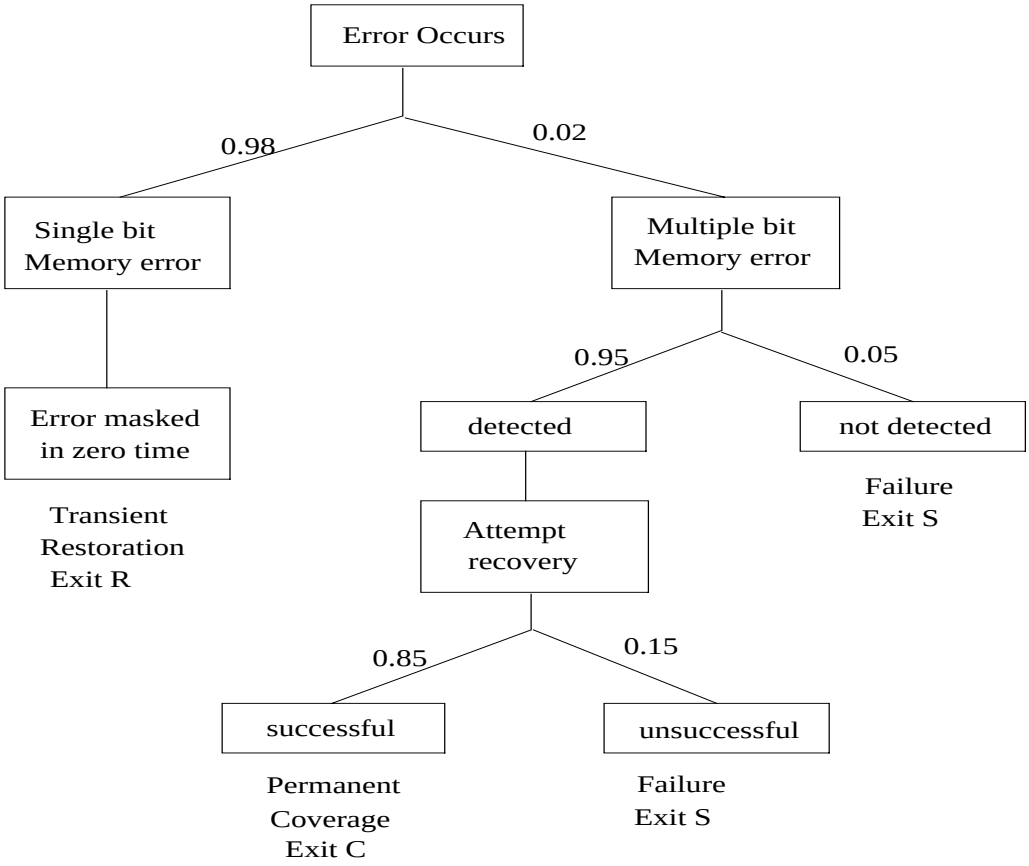


Figure 13-9. Coverage Model for Memory System

The 2% of faults that affect more than one memory bit are assumed to be 95% detectable. When a multiple memory error is detected, the affected portion of memory is discarded, the memory mapping function is updated, and the needed information is reloaded from a previous checkpoint and updated to represent the current state of the system. Experimentation on a prototype system revealed that this recovery from the detected multiple-memory errors works 85% of the time. Thus, the probability of reaching the **C** exit is the probability that a multiple fault occurs, is detected, and is recovered from is $c_M = (0.02) \bullet (0.95) \bullet (0.85) = 0.01615$.

There are two paths to the **S** (single point failure) exit. First, the memory fault causes a single-point failure if a multiple-bit error is not detected (with probability $0.02 \bullet 0.05$). Second, a single-point failure occurs if a multiple-bit memory error is detected, but the attempted recovery is not successful. Thus, $s_M = (0.02) \bullet ((0.05) + (0.95) \bullet (0.15)) = 0.00385$.

Memory interface units

Experience with memory interface units suggests that 95% of all faults are recoverable transients, thus r_{MIU} = 0.95. Of the remaining 5% of faults, which are permanent, 80% are recoverable by discarding the affected M_{IU}. Thus, c_{MIU} = 0.4 and s_{MIU} = 0.1.

Bus, application and interface

For the remaining components (the bus, the application HW and SW and the human operator) it is assumed that all faults are permanent and are perfectly covered, thus $c = 1$ and $s = r = 0$.

13.3 Analysis Results

The HECS example system was analyzed using Galileo/ASSAP. Galileo/ASSAP analyzed the HECS fault tree by automatically dividing it into four subtrees, along the same lines as was done in building the tree. The results at the subtree level are shown in Figure 13-10, for a 100-hour mission. The processing system and the memory systems were solved as Markov models, while the rest was solved using static (BDD) methods. Importance analysis (using Birnbaum's importance measure*) results are also shown. Three of the subsystems are about equally important, and a small decrease in unreliability in these subsystems translates to an almost equal decrease in HECS unreliability (since the importance measures are close to 1). In terms of the basic components, the memory units, the memory interface units, the application SW and HW and human operator all have about the same importance (between 0.89 and 0.96). The processors are less important (0.66). The cold spare (0.002) and bus (0.0002) are not major contributors to unreliability.

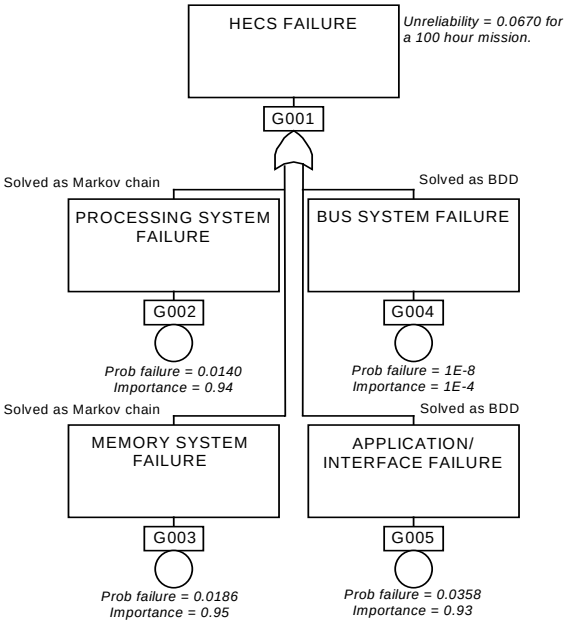


Figure 13-10. Analysis of HECS for 100-hour mission

* See Chapter 7 for a definition of this term.

For a 100-hour mission, the unreliability of the HECS system was found to be 0.067, which means that it is approximately 93% reliable for 100 hours. Figure 13-11 shows the unreliability for the HECS system for a period of up to 1000 hours (non-maintained).

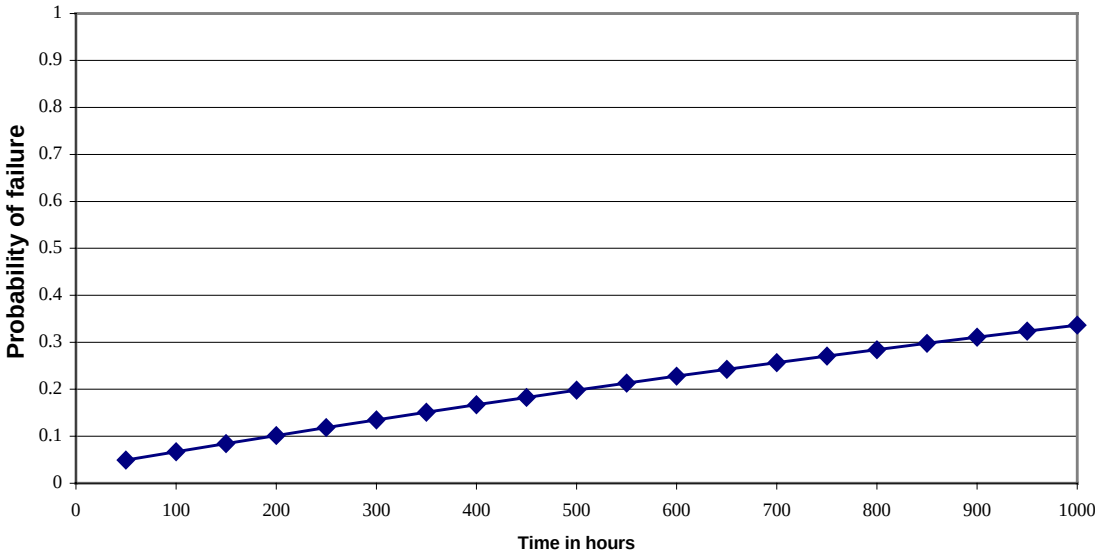


Figure 13-11. Unreliability for HECS

APPENDICES

Appendix A. Boolean Algebra And Application To Fault Tree Analysis

A.1 Rules of Boolean Algebra

The Boolean techniques discussed in this appendix have immediate practical importance in relation to fault trees. A fault tree can be thought of as a pictorial representation of those Boolean relationships among fault events that cause the top event to occur. In fact, a fault tree can always be translated into an entirely equivalent set of Boolean equations. Thus an understanding of the rules of Boolean algebra contributes materially toward the construction and simplification of fault trees. Once a fault tree has been drawn, it can be evaluated to yield its qualitative and quantitative characteristics. These characteristics cannot be obtained from the fault tree per se, but they can be obtained from the equivalent Boolean equations. In this evaluation process the algebraic reduction techniques discussed in this appendix are used.

The rules of Boolean algebra are presented in Table A-1 along with a short discussion of each rule. The reader is urged to check the validity of each rule by recourse to Venn diagrams, as illustrated in Appendix B, Section 1. Those readers who are mathematically inclined will detect that the rules, as stated, do not constitute a minimal necessary and sufficient set. Here and elsewhere, mathematical elegance has sometimes been sacrificed in favor of a more useful and understandable presentation for the practical system analyst.

According to relations (la) and (lb), the union and intersection operations are commutative. In other words, the commutative laws permit interchange of the events X, Y with regard to an "AND" or "OR" operation. It is important to remember that there are mathematical entities that do not commute; e.g. the vector cross product and matrices in general.

Relations (2a) and (2b) are similar to the associative laws of ordinary algebra: a(bc) = (ab)c and a + (b+c) = (a+b) + c. If a series of "OR" operations or a series of "AND" operations are under consideration, the associative laws permits grouping the events any way desired.

The distributive laws, relations (3a) and (3b) provide the valid manipulatory procedure whenever a combination of an "AND" operation with an "OR" operation is under consideration. By proceeding from left to right in the equations, the left-hand expression is reduced to an unfactored form. In relation (3a), for example, operating with X on Y and on Z results in the right-hand expression. Operating from right to left in the equations simply factors the expression. For instance, in relation (3b) X is factored out to obtain the left-hand side. Although relation (3a) is analogous to the distributive law in ordinary algebra, relation (3b) has no such analog.

Table A-1. Rules of Boolean Algebra

	Mathematical Symbolism	Engineering Symbolism	Designation
(1a)	$X \cap Y = Y \cap X$	$X \bullet Y = Y \bullet X$	Commutative Law
(1b)	$X \cup Y = Y \cup X$	$X + Y = Y + X$	
(2a)	$X \cap (Y \cap Z) = (X \cap Y) \cap Z$	$X \bullet (Y \bullet Z) = (X \bullet Y) \bullet Z$ $X \bullet (Y \bullet Z) = (X \bullet Y) \bullet Z$	Associative Law
(2b)	$X \cup (Y \cup Z) = (X \cup Y) \cup Z$	$X + (Y + Z) = (X + Y) + Z$	
(3a)	$X \cap (Y \cup Z) = (X \cap Y) \cup (X \cap Z)$	$X \bullet (Y + Z) = X \bullet Y + X \bullet Z$ $X \bullet (Y + Z) = X \bullet Y + X \bullet Z$	Distributive Law
(3b)	$X \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z)$	$X + Y \bullet Z = (X + Y) \bullet (X + Z)$	
(4a)	$X \cap X = X$	$X \bullet X = X$	Idempotent Law
(4b)	$X \cup X = X$	$X + X = X$	
(5a)	$X \cap (X \cup Y) = X$	$X \bullet (X + Y) = X$	Law of Absorption
(5b)	$X \cup (X \cap Y) = X$	$X + X \bullet Y = X$	
(6a)	$X \cap X' = \phi$	$X \bullet X' = \phi$	Complementation
(6b)	$X \cup X' = \Omega = I^*$	$X + X' = \Omega = I$	
(6c)	$(X')' = X$	$(X')' = X$	
(7a)	$(X \cap Y)' = X' \cup Y'$	$(X \bullet Y)' = X' + Y'$	de Morgan's Theorem
(7b)	$(X \cup Y)' = X' \cap Y'$	$(X + Y)' = X' \bullet Y'$	
(8a)	$\phi \cap X = \phi$	$\phi \bullet X = \phi$	Operations with ϕ and Ω
(8b)	$\phi \cup X = X$	$\phi + X = X$	
(8c)	$\Omega \cap X = X$	$\Omega \bullet X = X$	
(8d)	$\Omega \cup X = \Omega$	$\Omega + X = \Omega$	
(8e)	$\phi' = \Omega$	$\phi' = \Omega$	
(8f)	$\Omega' = \phi$	$\Omega' = \phi$	
(9a)	$X \cup (X' \cap Y) = X \cup Y$	$X + X' \bullet Y = X + Y$	These relationships are unnamed but are frequently useful in the reduction process.
(9b)	$X' \cap (X \cup Y') = X' \cap Y' = (X \cup Y)'$	$X' \bullet (X + Y') = X' \bullet Y' = (X + Y)'$	

*The symbol I is often used instead of Ω to designate the Universal Set. In engineering notation Ω is often replaced by 1 and ϕ by 0.

The idempotent laws, relations (4a) and (4b), "cancel out" any redundancies of the same event.

The laws of absorption, relations (5a) and (5b), can easily be validated by reference to an appropriate Venn diagram. Relation (5a) can also be argued in the following way. Whenever the occurrence of X automatically implies the occurrence of Y, then X is said to be a subset of Y. This situation can be represented in symbolic form as $X \subset Y$ or $X \rightarrow Y$. In this case $X + Y = Y$ and $X \bullet Y = X$. In relation (5a), if X occurs then (X+Y) has also occurred and $X \subset (X+Y)$; therefore $X \bullet (X+Y) = X$. A similar argument can be developed in the case of relation (5b).

De Morgan's theorems, relations (7a) and (7b), provide the general rules for removing primes on brackets. Suppose that X represents the failure of some component. Then X' represents the non-

failure or successful operation of that component. In this light, relation (7a) simply states that for the double failure of X and Y not to occur, either X must not fail or Y must not fail.

As an application of the use of these rules, consider the simplification of the expression

$$(A+B) \bullet (A+C) \bullet (D+B) \bullet (D+C).$$

Applying relation (3b) to $(A+B) \bullet (A+C)$ results in

$$(A+B) \bullet (A+C) = A+(B \bullet C).$$

Likewise,

$$(D+B) \bullet (D+C) = D+(B \bullet C).$$

An intermediate result produced is

$$(A+B) \bullet (A+C) \bullet (D+B) \bullet (D+C) = (A+B \bullet C) \bullet (D+B \bullet C).$$

Letting E represent the event $B \bullet C$ results in:

$$(A+B \bullet C) \bullet (D+B \bullet C) = (A+E) \bullet (D+E) = (E+A) \bullet (E+D).$$

Another application of relation (3b) yields

$$(E+A) \bullet (E+D) = E+A \bullet D = B \bullet C + A \bullet D.$$

Therefore, the final result is

$$(A+B) \bullet (A+C) \bullet (D+B) \bullet (D+C) = B \bullet C + A \bullet D.$$

The original expression has been substantially simplified for purposes of evaluation.

A.2 Determining the Minimal Cut Sets or Minimal Path Sets of a Fault Tree

One of the main purposes of representing a fault tree in terms of Boolean equations is that these equations can then be used to determine the fault tree's associated minimal cut sets and minimal path sets. Once the minimal cut sets are obtained, the quantification of the fault tree is more or less straightforward. The minimal path sets are essentially the complements of the minimal cut sets and define the “success modes” by which the top event will not occur. The minimal path sets are often not obtained in a fault tree evaluation; however, they can be useful in particular problems.

Minimal Cut Sets

By the definition, a minimal cut set is a combination (intersection) of primary events sufficient for the top event. The combination is a “minimal” combination in that all the failures are needed for the top event to occur; if one of the failures in the cut set does not occur, then the top event will not occur (by this combination).

Any fault tree will consist of a finite number of minimal cut sets that are unique for that top event. One-component minimal cut sets, if there are any, represent those single failures that will cause the top event to occur. Two-component minimal cut sets represent the double failures that together will cause the top event to occur. For an n-component minimal cut set, all n components in the cut set must fail in order for the top event to occur.

The minimal cut set expression for the top event can be written in the general form,

$$T = M_l + M_2 + \ldots + M_{k,}$$

where T is the top event and M_i are the minimal cut sets. Each minimal cut set consists of a combination of specific component failures, and hence the general n-component minimal cut can be expressed as

$$M_i = X_1 \bullet X_2 \bullet \ldots \bullet X_n,$$

where X_1, X_2, etc., are basic component failures in the tree. An example of a top event expression is

$$T = A + B \bullet C,$$

where A, B, and C are component failures. This top event has a one-component minimal cut set (A) and a two-component minimal cut set (B•C). The minimal cut sets are unique for a top event and are independent of the different equivalent forms the same fault tree may have.

To determine the minimal cut sets of a fault tree, the tree is first translated to its equivalent Boolean equations. A variety of algorithms exist to translate the Boolean equations into cut sets. Two of the most common are the "top-down" or "bottom-up" substitution methods to solve for the top event. The methods are straightforward and involve substituting and expanding Boolean expressions. The distributive law, relations (3a) and (3b) in Section A.1, and the law of absorption, relations (5a) and (5b), are used to remove the redundancies.

Consider the simple fault tree shown in Figure A-1; the equivalent Boolean equations are shown following the tree.

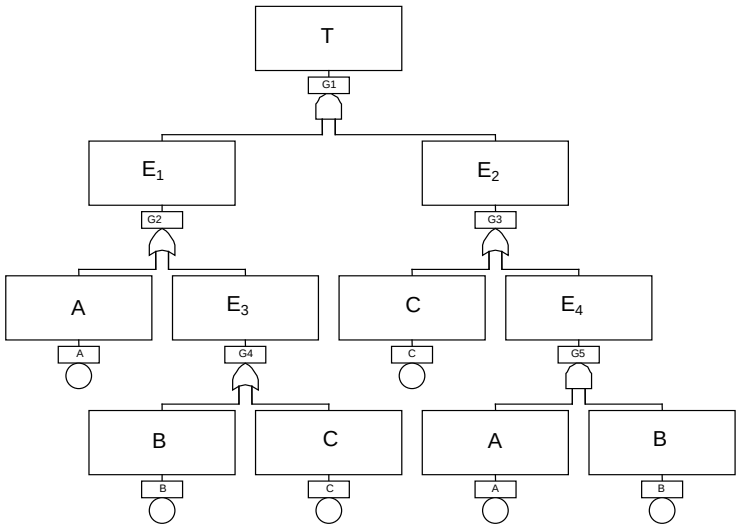


Figure A-1. Example Fault Tree

$$T = E_1 \bullet E_2$$
$$E_1 = A + E_3$$
$$E_3 = B + C$$
$$E_2 = C + E_4$$
$$E_4 = A \bullet B$$

First the top-down substitution will be performed, starting with the top event equation and substituting and expanding until the minimal cut set expression for the top event is obtained. Substituting for E_1 and E_2 and expanding produces

$$\begin{aligned} T &= (A+E_3) \bullet (C+E_4) \\ &= (A \bullet C) + (E_3 \bullet C) + (E_4 \bullet A) + (E_3 \bullet E_4). \end{aligned}$$

Substituting for E_3,

$$\begin{aligned} T &= A \bullet C + (B+C) \bullet C + E_4 \bullet A + (B+C) \bullet E_4 \\ &= A \bullet C + B \bullet C + C \bullet C + E_4 \bullet A + E_4 \bullet B + E_4 \bullet C. \end{aligned}$$

By the idempotent law, $C \bullet C = C$, which results in

$$T = A \bullet C + B \bullet C + C + E_4 \bullet A + E_4 \bullet B + E_4 \bullet C.$$

But $A \bullet C + B \bullet C + C + E_4 \bullet C = C$ by the law of absorption. Therefore,

$$T = C + E_4 \bullet A + E_4 \bullet B.$$

Finally, substituting for E_4 and applying the law of absorption twice

$$\begin{aligned} T &= C + (A \bullet B) \bullet A + (A \bullet B) \bullet B. \\ &= C + A \bullet B. \end{aligned}$$

The minimal cut sets of the top event are thus C and A•B; one single component minimal cut set and one double component minimal cut set. The fault tree can thus be represented as shown in Figure A-2, which is equivalent to the original tree (both trees have the same minimal cut sets).

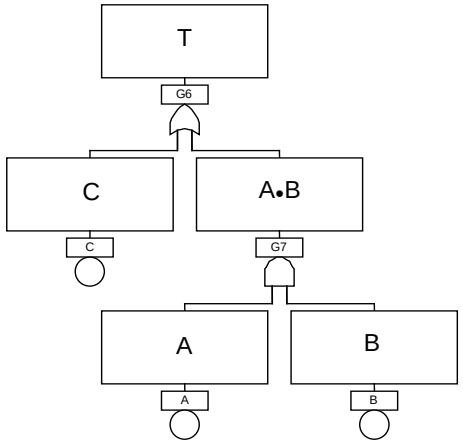


Figure A-2. Fault Tree Equivalent of Figure A-1.

The bottom-up method uses the same substitution and expansion techniques, except that now the operation begins at the bottom of the tree and proceeds upward. Equations containing only basic failures are successively substituted for higher faults. The bottom-up approach can be more laborious and time-consuming; however, the minimal cut sets are now obtained for every intermediate fault as well as for the top event.

Consider again the example tree (the equivalent Boolean equations are repeated for the reader's convenience).

$$T = E_1 \bullet E_2$$
$$E_1 = A + E_3$$
$$E_3 = B + C$$
$$E_2 = C + E_4$$
$$E_4 = A \bullet B$$

Because E_4 involves only basic failures, it is substituted into E_2 to obtain

$$E_2 = C + A \bullet B.$$

The minimal cut sets of E_2 are thus C and A•B. E_3 is already in reduced form having minimal cut sets B and C. Substituting into E_1, $E_1 = A+B+C$ is obtained. E_1 therefore has three minimal cut sets A, B, and C. Finally, substituting the expressions for E_1 and E_2 into the equation for T, expanding and applying the absorption law, results in

$$\begin{aligned} T &= (A+B+C) \bullet (C+A \bullet B) \\ &= A \bullet C + A \bullet A \bullet B + B \bullet C + B \bullet A \bullet B + C \bullet C + C \bullet A \bullet B \\ &= A \bullet C + A \bullet B + B \bullet C + A \bullet B + C + A \bullet B \bullet C \\ &= C + A \bullet B. \end{aligned}$$

The minimal cut sets of the top event are thus again C and A•B.

As a very simple example, assume the pumping system shown in Figure A-3.

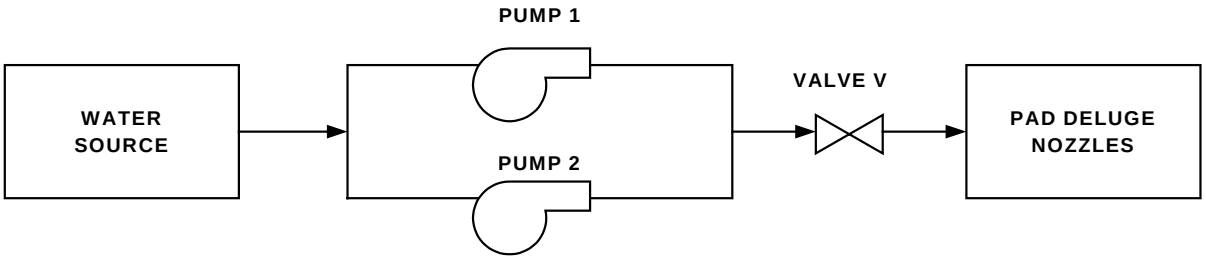


Figure A-3. Water Pumping System

Assume that the undesired event is "no flow of water to pad deluge nozzles." Ignoring the contribution of the pipes, this system can be modeled by the fault tree of Figure A-2, where:

T = "no flow of water to pad deluge nozzles"
C = "valve V fails closed"
A = "pump 1 fails to run"
B = "pump 2 fails to run"

It has just been shown that the minimal cut sets of this tree are C and A•B. The undesired event "no flow of water to tank" will therefore occur if either valve V fails closed or both pumps fail to run. In this simple case, the cut sets do not really provide any insights that are not already quite obvious from the system diagram. In more complex systems, however, where the system failure modes are not so obvious, the minimal cut set computation provides the analyst with a thorough and systematic method for identifying the basic combinations of component failures that can cause an undesired event.

For small fault trees, the determination of the minimal cut sets, using either the top-down or bottom-up method, can be done by hand. For larger trees, various computer algorithms and software for fault tree evaluation are available.

Minimal Path Sets and Compliment Fault Trees

The top event of a fault tree often represents system failure. System failure is of great interest from the point of view of system safety. However, from the point of view of reliability the concern would more likely be with the prevention of the top event. Since the top event of a fault tree can be represented by a Boolean equation, and because this equation can be complemented, there is also a Boolean equation for the complement (i.e., nonoccurrence) of the top event. This complemented equation, in turn, corresponds to a tree that is the complement of the original tree. This complemented tree may be obtained directly from the original tree by complementing all the events and substituting OR-gates for AND-gates and vice versa. In either case, whether the top event equation or the tree itself is complemented, de Morgan's theorem as given in Table A-1 is applied. The minimal cut sets of the dual tree are the so-called "minimal path sets" of the

original tree—the smallest combination (intersection) of primary events whose non-occurrence assures the non-occurrence of the top event.

The combination is a smallest combination in that all the primary event nonoccurrences are needed for the top event to not occur; if any one of the events occurs then the top event can potentially occur. The minimal path set expression for the top event T can be written as

$$T' = P_1 + P_2 + \ldots + P_k$$

where T' denotes the complement (nonoccurrence) of T. The terms P_1, P_2, … P_k are the minimal path sets of the fault tree. Each path set can be written as

$$P_i = X'_1 \bullet X'_2 \bullet \ldots \bullet X'_m,$$

where X_i represents the basic events in the fault tree and X'_i represents the complements.

The minimal path sets of a given fault tree can be found by forming its dual and then using either the top-down or bottom-up method to find the dual's minimal cut sets. These cut sets are the minimal path sets of the original tree which were desired.

Alternatively, if the minimal cut sets of the fault tree have already been determined, the complement of the minimal cut set equation can be generated and the minimal path sets obtained directly. For the sample tree, the following minimal cut expression was previously obtained:

$$T = C + A \bullet B.$$

Taking the complement using de Morgan's theorem

$$\begin{aligned} T' &= (C+A \bullet B)' \\ &= C' \bullet (A \bullet B)'. \end{aligned}$$

Applying de Morgan's theorem to the term (A•B)',

$$T' = C' \bullet (A'+B');$$

and using the distributive law (i.e., expanding)

$$T' = C' \bullet A' + C' \bullet B'.$$

Therefore, the minimal path sets of the tree are C'•A' and C'•B'. In terms of the pumping system of Figure A-3, this indicates that the undesired event can be prevented and the system assured success if either:

(1) Valve V is open and pump 1 is running, or
(2) Valve V is open and pump 2 is running.

Appendix B. Probability Theory: The Mathematical Description Of Events

This Appendix addresses the basic mathematical technique involved in the quantitative assessment of fault trees: probability theory. Probability theory is basic to fault tree analysis because it provides an analytical treatment of *events*, and events are the fundamental components of fault trees. The topics that will be considered include set theory, the algebra of probabilities, and Bayes theorem, all of which provide a foundation for the treatment of events and the quantification of probabilities of events.

B.1 Set Theory—Application to the Mathematical Treatment of Events

As seen in the previous section, combinatorial analysis allows for the determination of the number of combinations pertinent to an event of interest. Set theory is a more general approach that allows the outcome events of an experiment to be "organized" to determine the appropriate probabilities. In the most general sense, a set is a collection of items having some recognizable features in common so that they may be distinguished. Examples are prime numbers, relays, rocket engines, solutions of Bessel's equation, etc. The application of set theory addressed here involves a considerable particularization. The items of immediate interest are the outcome events of random experiments and the development of the elementary notions of set theory will be restricted to the event concept.

An event can be thought of as a collection of elements. Consider, for example, the following possible events of interest associated with the toss of a die:

A - the number 2 turns up
B - the result is an even number
C - the result is less than 4
D - some number turns up
E - the result is divisible by 7.

Each of these events can be considered as a particular set whose elements are drawn from the basic outcome space of the experiment: {l, 2, 3, 4, 5, 6}.

That is:

A = {2}
B = {2, 4, 6}
C = {1, 2, 3}
D = {l, 2, 3, 4, 5, 6}
E = ϕ (the null, void, or empty set),

where braces, "{ }," are used to denote a particular set and the quantities within the braces are the elements of that set.

Event A can be represented as a set having a single element—the number 2. Both B and C can be represented as three-element sets. Event D contains all the possible results of the experiment. It thus coincides with the outcome space. Any such set that contains all the outcomes of an

experiment is referred to as the *universal set* and is generally denoted by the symbol Ω or I (also sometimes by the number 1 when the notation is informal). E is an impossible event and can be represented by a set containing no elements at all, the so-called *null set* symbolized by ϕ.

Returning to the die-throwing example, note that the element "1" belongs to C and D but not to A or B. This fact is symbolized as follows:

$$1 \in C, 1 \in D, 1 \notin A, 1 \notin B,$$

where the symbol "$\in$" means "is an element of" and the symbol $\notin$ means "is not an element of."

Note also that the elements of A, B, C are contained in set D. A, B, C are therefore called subsets of D and this can be expressed by $A \subset D$, $B \subset D$, $C \subset D$. Observe also that A is a subset of both B and C. If X and Y are two sets such that X is a subset of Y, i.e., $X \subset Y$ and Y is a subset of X, i.e., $Y \subset X$, then X and Y are equal (i.e., they are the same set).

As another example, consider the time of failure t of a diesel (in hours) and consider the sets

$$A = \{t=0\}$$
$$B = \{t_i,\ 0 < t \leq 1\}$$
$$C = \{t_j,\ t > 1\}.$$

The failure to start of the diesel is represented by A, i.e., "zero failure time." B represents times of failure which are greater than zero hours (i.e., the diesel started) and less than or equal to one hour. C represents time of failure greater than 1 hour. Each of these sets, i.e., events, could be associated with different consequences if an abnormal situation existed (e.g., loss of utility power).

There exists a graphical representation that permits a simple visualization of the set theory concepts. This representation is known as a *Venn diagram*. The universal set is represented by some geometrical shape (usually a rectangle) and any subsets (events) of interest are shown inside. Figure B-1 represents a Venn diagram for the previous toss-of-a-die example.

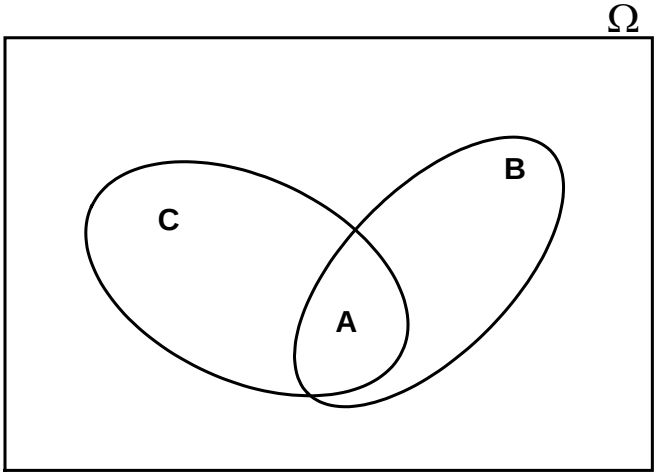


Figure B-1. Venn Diagram Representation of Sets

Operations on sets (events) can be defined with the help of Venn diagrams. The operation of union is portrayed in Figure B-2.

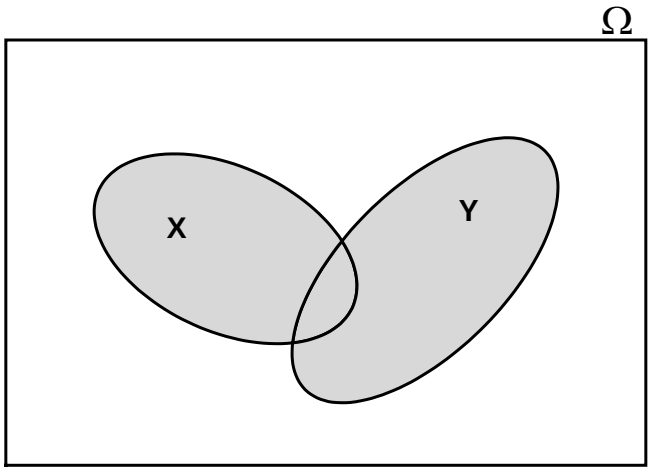


Figure B-2. The Operation of Union

The union of two sets of X, Y is the set that contains all elements that are either in X or in Y or in both, is written as X∪Y, and is indicated by the shaded area in Figure B-2. Returning to the die example, the union of B, C is written,

$$B \cup C = \{1, 2, 3, 4, 6\}$$

Note that the word "or" translates into the symbol "∪."

The operation of *intersection* is portrayed in Figure B-3. The intersection of two sets X, Y is the set that contains all elements that are common to X and Y, is written as X∩Y, and is indicated by the shaded area in Figure B-3. In the die example, $B \cap C = \{2\} = A$. Note that the word "and" translates into the symbol "∩."

The operation of *complementation* portrayed in Figure B-4. The complement of a set X is the set that contains all elements that are not in X, is written $\overline{X}$ (or X'), and is indicated by the shaded area in Figure B-4. For the die example, the complement of the set $B \cup C$ is $(B \cup C)' = \overline{(B \cup C)} = \{5\}$.

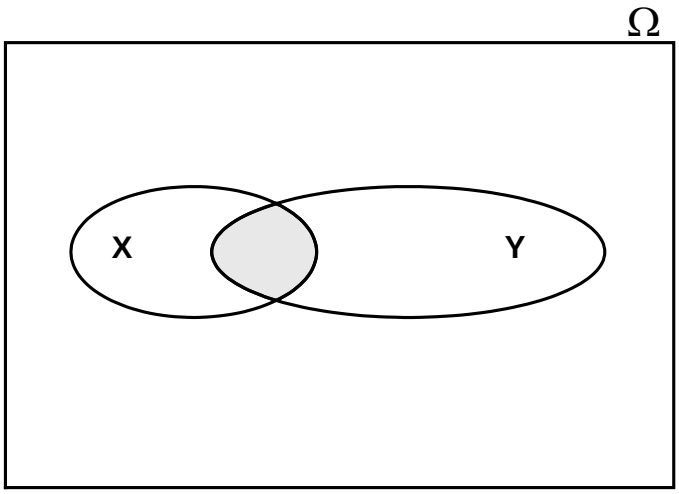


Figure B-3. The Operation of Intersection

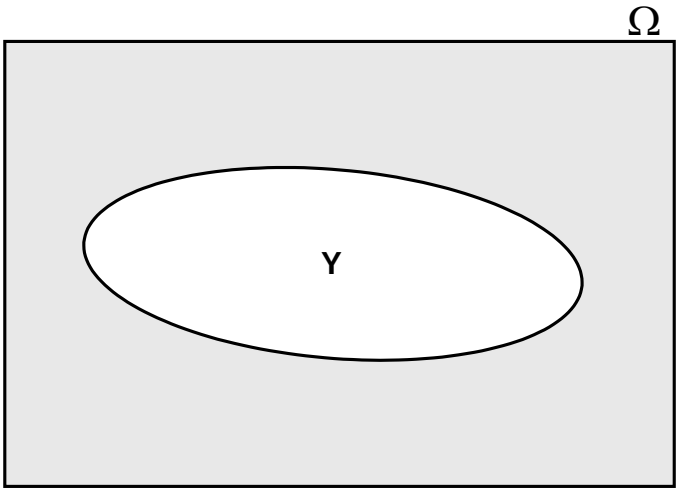


Figure B-4. The Operation of Complementation

There is another operation (unnamed) that is sometimes defined, but it is not independent of the operations that have already been given. This operation is illustrated in Figure B-5. If all elements are removed from set Y that are common to both X and Y, the shaded area indicated in Figure B-5 is what is left. This process is occasionally written (Y-X) but the reader can readily see that

$$(Y - X) = Y \cap X',$$

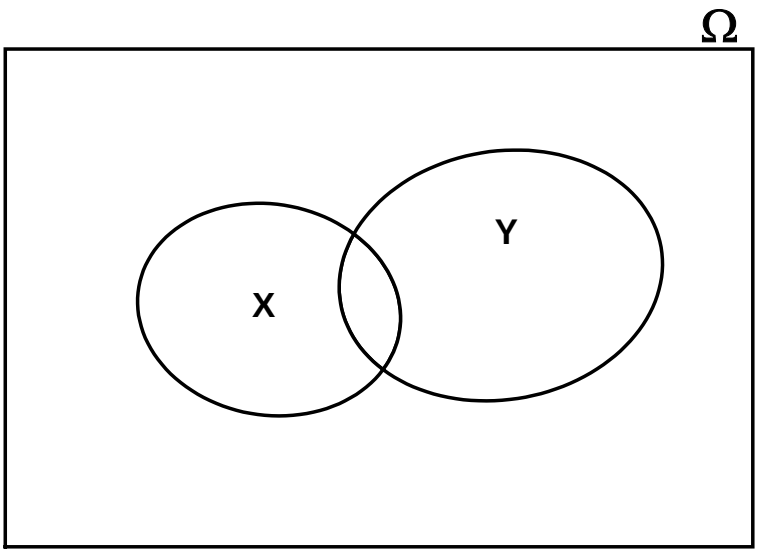


Figure B-5. The Operation (Y-X)

so that the symbol (Y-X) is not needed and hence will not be further used.

As an example, consider a simple system consisting of three components A, B, C. Consider using the symbols A, B, C not only to designate the components themselves but also the events "component success" for A, B, C, respectively. Thus, $\overline{A}, \overline{B}, \overline{C}$, represent the events "component failure" of A, B, C, respectively. The symbol $A\overline{B}C$ will consequently be used to represent the event, "A operates successfully, B fails, and C operates successfully."

Because there are three components and two modes of operation for each component (success/failure), combinatorial analysis tells us that there are $2^3 = 8$ combinations which give all modes of failure or operation for the system. Thus, the universal set (or outcome collection) is given by:

$$\Omega = \left\{ABC, AB\overline{C}, A\overline{B}C, A\overline{BC}, \overline{A}BC, \overline{A}B\overline{C}, \overline{AB}C, \overline{ABC}\right\}.$$

Suppose it is determined that the system will fail if any two or more of its components fail. Then the events corresponding to system failure are:

$$S_1 = A\overline{BC}$$
$$S_2 = \overline{A}B\overline{C}$$
$$S_3 = \overline{AB}C$$
$$S_4 = \overline{ABC}$$

and the event, "system failure," S, can be represented by the subset $S = S_1 \cup S_2 \cup S_3 \cup S_4 = \left\{A\overline{BC}, \overline{A}B\overline{C}, \overline{AB}C, \overline{ABC}\right\}$. All the ways the system can fail have been enumerated in an exhaustive manner. This information can be used in various applications. For example, if the

probability of component failures is known, the probability of system failure can be calculated. In the same way that has been illustrated above, intersections and unions of any of the basic elements (simple events) of the universal set may be taken to generate still other events that can be represented as sets consisting of particular elements.

Using the set theory concepts that have been developed, probability equations can be translated into set-theoretic terms. For example,

$$P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B) \text{ becomes}$$
$$P(A \cup B) = P(A) + P(B) - P(A \cap B). \quad \text{(B.1)}$$

The equation

$$P(A \text{ and } B) = P(A|B)\,P(B) = P(B|A)\,P(A) \text{ becomes}$$
$$P(A \cap B) = P(A|B)\,P(B) = P(B|A)\,P(A). \quad \text{(B.2)}$$

B.2 Symbolism

Boolean algebra, which is the algebra of events, deals with event operations that are represented by various symbols. Unfortunately, symbolism in set theory is not uniform; the symbology differs among the fields of mathematics, logic, and engineering as follows:

Operation	Probability	Mathematics	Logic	Engineering
Union of A and B	A or B	$A \cup B$	$A \vee B$	A+B
Intersection of A and B	A and B	$A \cap B$	$A \wedge B$	A•B or AB
Complement of A	not A	A' or $\overline{A}$	-A	A' or $\overline{A}$

The symbols used in mathematics and in logic are very similar. The logical symbols are the older; in fact the symbol "$\vee$" is an abbreviation for the Latin word "vel" which means "or." It is unfortunate that engineering has adopted "+" for "$\cup$" and an implied multiplication for "$\cap$" This procedure "overworks" the symbols "+" and "•". As an example of the confusion that might arise when "+" is used for $\cup$, consider the expression

$$P(A \cup B) = P(A) + P(B) - P(A \cap B).$$

If "+" is written for "$\cup$" on the left-hand side, an equation with "+" meaning one thing on the left and a totally different thing on the right is produced.

Despite these difficulties and potentially confusing elements in the symbology, the engineering symbology is now quite widespread in the engineering literature and any expectation of a return to mathematical or logic symbols seems futile. In fault tree analysis, use of the engineering notation is widespread, and, as a matter of fact, it is used elsewhere in this handbook. If, however, the reader is unacquainted with event algebra, it is strongly recommended that the proper mathematical symbols be used until attaining familiarity with this type of algebra. This

will serve as a reminder that set algebraic operations are not to be confused with the operations of ordinary algebra where numbers, and not events, are manipulated.

B.3 Additional Set Concepts

Certain addition set concepts will now be presented. These will illustrate the difference between *simple events* and *compound events*. This distinction is useful for some fault tree concepts and also leads to a more rigorous definition of "probability."

Consider again the throw of a single die. The event A = {2} is a simple event; in fact it constitutes an element of the outcome space. In contrast, the events B = {2, 4, 6} and C = {1, 2, 3} are compound events. They do not constitute, per se, elements of the outcome space, even though they are made up of elements of the outcome space. Events B and C have an element in common; therefore, their intersection is non-empty (i.e., they are not "disjoint" subsets or, in probability language, they are not mutually exclusive). The elements of outcome collections are, by definition, all mutually exclusive and, thus, all mutually disjoint.

Compound events (like B and C) are generally the ones that are of predominant interest in the real world and it is necessary, because they are not included in the outcome space, to define a mathematical entity that does include them. Such a mathematical entity is called a class. A class is a set whose elements are themselves sets and these elements are generated by enumerating every possible combination of the members of the original outcome space.

As an example, consider the four-element outcome space S = {l, 3, 5, 7}. If every possible combination of these four elements is listed the class $\underline{S}$ is generated and is defined on the original set S as follows:

$$\underline{S} = \{1\}, \{3\}, \{5\}, \{7\}, \{1, 3\}, \{1, 5\},$$
$$\{l, 7\}, \{3, 5\}, \{3, 7\}, \{5, 7\}, \{1, 3, 5\}$$
$$\{1, 5, 7\}, \{1, 3, 7\}, \{3, 5, 7\}, \{1, 3, 5, 7\}, \{\phi\}.$$

Notice that the null set ϕ is considered an element of the class to provide a mathematical description of the "impossible event." If the number of elements in the class $\underline{S}$ are counted, the total is determined to be 16, which is 2^4, where 4 is the number of elements in the original set S. In general, if the original set has n elements, the corresponding class will have 2^n elements.

The utility of the class concept is simply that the class will contain as elements, every conceivable result (both simple and compound) of an experiment. Thus, in the die toss experiments, S will have 6 elements and $\underline{S}$ will have $2^6 = 64$ elements, two of which will be B = {2, 4, 6} and C = {1, 2, 3}. In the throw of 2 dice, S will contain 36 elements and $\underline{S}$ will have 2^{36} (a number in excess of 10^{10}) elements. Embedded somewhere in this enormous number of subsets the compound event "sum = 7" is formed and it can be represented in the following way:

$$E = \{(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)\}.$$

Consider again the simple three-component system A, B, C, where system failure consists of any two or more components failing. In that example, the universal set comprised eight elements,

and these eight elements gave all modes of failure or operation of the system. The class based on this set would contain $2^8 = 256$ elements and would include such events as

"A operates properly": $\{ABC, AB\overline{C}, A\overline{B}C, A\overline{B}\,\overline{C}\}$,
"Both B and C fail": $\{A\overline{B}\,\overline{C}, \overline{A}\,\overline{B}\,\overline{C}\}$,
"Five components fail": ϕ, and
"System fails": $\{A\overline{B}\,\overline{C}, \overline{A}B\overline{C}, \overline{A}\,\overline{B}C, \overline{A}\,\overline{B}\,\overline{C}\}$.

The reader should bear in mind that the elements of classes are *sets*. Thus, the event ABC is an element of the original universal set, whereas {ABC} is a set containing the single element ABC and is an element of the class generated from the universal set. The utility of the class concept is that it enables compound events to be treated in a formal manner simply because all possible compound events are included in the class.

Perhaps the most useful feature of the class concept is that it provides a basis for establishing a proper mathematical definition of the probability function. The set theoretic definition of the probability function is shown schematically in Figure B-6.

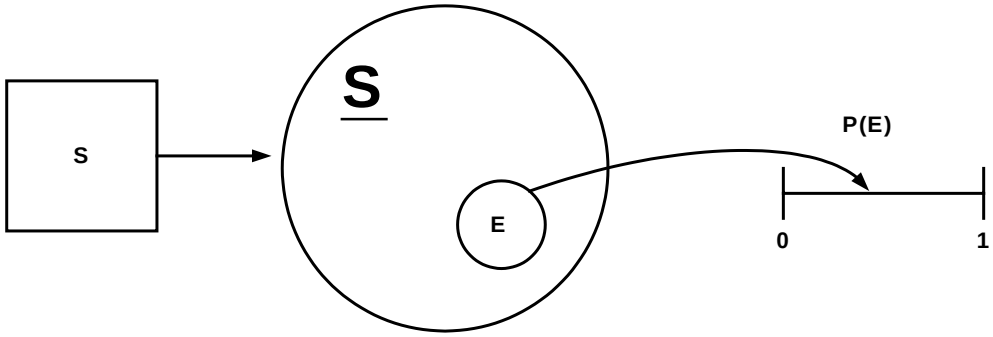


Figure B-6. Set Theoretic Definition of Probability

In Figure B-6, the box labeled "S" represents the outcome collection for some random experiment. It could, for instance, represent the totality of the 36 possible outcomes in the two-dice experiment. The circle labeled "S" represents the *class* generated from set S by enumerating all combinations of the elements of S. In the two-dice example the class S possesses an enormous number ($> 10^{10}$) of elements. These elements represent every conceivable outcome (simple and compound) of the experiment. Specifically, the event E = "sum of 7" is a member of S. Event E is shown schematically in Figure B-6. Next, the axis of real numbers between 0 and 1 is drawn. A function can now be defined that "maps" event E into some position on this axis. This function is the probability function P(E).

The concept of mapping may be unfamiliar. For the purpose of this handbook a mapping may be considered simply as establishing a functional relationship. For instance, the relation $y = x^2$ maps all numbers x into a parabola ($x = \pm 1$, $y = +1$; $x = \pm 2$, $y = +4$; etc.). The relation $y = x$ maps all numbers x into a straight line making a 45° angle with the y-axis. In these examples one range of numbers is mapped into another range of numbers and the examples are referred to as *point-functions*. The function P(E) is somewhat more complicated; it maps a set onto a range of

numbers, and is referred to as a *set function* instead of a point function although the underlying concept is the same. In unsophisticated terms, however, a probability function simply assigns one unique number, a probability, to each event.

Two things should be noted. One is that probability has now been defined without making use of the limit of a ratio (e.g., as the ratio of the number of favorable outcomes to the total number of outcomes). The second thing is that this definition does not provide a method for calculating probability; rather, it explains the mathematical nature of the probability functions. If E is the event "sum = 7," the method to be used to calculate its probability assuming that all 36 outcomes in the outcome collection are equally likely has been discussed previously:

$$P(E) = \frac{6}{36} = \frac{1}{6}.$$

Of course this is a particularly simple example. In other cases the physical nature of the problem may have to be investigated, sometimes in great detail, in order to develop the probabilities of events of interest.

B.4 Algebraic Operations with Probabilities

Consider a random experiment and designate two of its possible outcomes as A and B. If A and B are *mutually exclusive*, then A and B cannot both happen on a single trial of the experiment. For instance, either Heads or Tails is expected as a result of a coin toss. Both Heads and Tails occurring on a single toss is not possible. If events A and B are mutually exclusive, an expression for the probability that either A or B occurs can be written:

$$P(A \text{ or } B) = P(A) + P(B). \quad (B.3)$$

This relation is sometimes referred to as the *addition rule for probabilities* and is applicable to events that are mutually exclusive. This formula can be readily extended to any number of mutually exclusive events A, B, C, D, E, ... :

$$P(A \text{ or } B \text{ or } C \text{ or } D \text{ or } E \text{ or } \ldots) = P(A) + P(B) + P(C) + P(D) + P(E) + \ldots \quad (B.4)$$

For events that are not mutually exclusive a more general formula must be used. For example, suppose that the random experiment is the toss of a single die in which two events are defined as follows:

A = "the number 2 turns up"
B = "an even number turns up."

Clearly these events are not mutually exclusive because if the result of the toss is 2, both A and B have occurred. The general expression for P(A or B) is now

$$P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B) \quad (B.5)$$

If A and B are mutually exclusive, P(A and B) = 0 and Equation B.5 reduces to Equation B.3. The reader should also note that Equation B.3 always gives an upper bound to the true probability

Equation B.5 when the events are not mutually exclusive. Now, returning to the single die problem and defining A and B as above, P(A or B) can be calculated numerically:

$$P(A \text{ or } B) = 1/6 + 1/2 - 1/6 = ½.$$

Equation B.5 can be extended to any number of events. For example, for three events A, B, C

$$P(A \text{ or } B \text{ or } C) = P(A) + P(B) + P(C) - P(A \text{ and } B) - P(A \text{ and } C) - P(B \text{ and } C) + P(A \text{ and } B \text{ and } C). \quad \text{(B.6)}$$

For n events E_1, E_2, …, E_n the general formula can be expressed as

$$\begin{aligned} P(E_1 \text{ or } E_2 \ldots \text{or } E_n) = & \sum_{i=1}^{n} P(E_i) - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P(E_i \text{ and } E_j) \\ & + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} P(E_i \text{ and } E_j \text{ and } E_k) \ldots \\ & + (-1)^n P(E_1 \text{ and } E_2 \text{ and} \ldots \text{and } E_n), \end{aligned} \quad \text{(B.7)}$$

where "Σ" is the summation sign.

If the possibility of any two or more of the events E_i occurring simultaneously is ignored, Equation B.7 reduces to

$$P(E_1 \text{ or } E_2 \text{ or} \ldots \text{or } E_n) = \sum_{i=1}^{n} P(E_i). \quad \text{(B.8)}$$

Equation B.8 is the so-called *rare event approximation* and is accurate to within about ten percent of the true probability when $P(E_i) < 0.1$. Furthermore, any error made is conservative in that the true probability is slightly lower than that given by Equation B.8.

Consider now two events A and B that are mutually independent. This means that in the course of several repetitions of the experiment, the occurrence (or non-occurrence) of A has no influence on the subsequent occurrence (or nonoccurrence) of B and vice versa. If a well-balanced coin is tossed randomly, the occurrence of Heads on the first toss should not cause the probability of Tails on the second toss to be any different from ½. The results of successive tosses of a coin are considered to be mutually independent outcomes. Also, if two components are operating in parallel and are isolated from one another, then the failure of one does not affect the failure of the other. In this case the failures of the components are independent events. If A and B are two mutually independent events, then

$$P(A \text{ and } B) = P(A)\, P(B). \quad \text{(B.9)}$$

This is often called the *multiplication rule for probabilities* and its extension to more than two events is obvious:

$$P(\text{A and B and C and D}) = P(A)\,P(B)\,P(C)\,P(D). \tag{B.10}$$

Very often, events that are not mutually independent are encountered, that is, they are mutually interdependent. For instance, the overheating of a resistor in an electronic circuit may very well change the failure probability of a nearby transistor or of related circuitry. The probability of rain on Tuesday will most likely be influenced by the weather conditions prevailing on Monday. In order to treat events of this nature, the concept of conditional probability is introduced, and a new symbol P(B|A) is needed that is the probability of B, given A has already occurred. The probability of A and B both occurring then becomes

$$P(\text{A and B}) = P(A)\,P(B|A) = P(B)\,P(A|B) \tag{B.11}$$

If A and B are mutually independent, then P(A|B) = P(A) and P(B|A) = P(B) and Equation B.11 reduces to Equation B.9. Thus, Equation B.11 constitutes a general expression for the probability of the joint occurrence of two events A and B.

For three events A, B, C,

$$P(\text{A and B and C}) = P(A)\,P(B|A)\,P(C|\text{A and B}), \tag{B.12}$$

where P(C|A and B) is the probability of C occurring given A and B have already occurred. For n events E_1, E_2, …, and E_n,

$$P(E_1 \text{ and } E_2 \text{ and } \ldots E_n) = P(E_1)\,P(E_2|E_1)\,P(E_3|E_1 \text{ and } E_2) \ldots P(E_n|E_1 \text{ and } E_2 \ldots \text{and } E_{n-1}). \tag{B.13}$$

An example of the use of the formulae in this section may be helpful. Consider the following random experiment: mentally select one card at random from a well-shuffled regulation deck of 52 cards. Note its face value (e.g., “seven,” “Queen,” etc.) and lay it aside (if the card is not put back in the deck this is known as *sampling without replacement*). Then choose a second card at random from the deck of 51 cards and note its face value. Now calculate the probability of getting at least one Ace in the two draws. There are three mutually exclusive possibilities:

(Ace on first draw then non-Ace on second draw) or
(non-Ace on first draw then Ace on second draw) or
(Ace on first draw then Ace again on second draw).

Expressed in mathematical language, this becomes much more succinct:

$$\begin{aligned} P(\text{at least one Ace in two draws}) &= P(A) \\ &= P(A_1 \text{ and } \bar{A}_2) + P(\bar{A}_1 \text{ and } A_2) + P(A_1 \text{ and } A_2) \\ &= P(A_1)\,P(\bar{A}_2|A_1) + P(\bar{A}_1)\,P(A_2|\bar{A}_1) + P(A_1)\,P(A_2|A_1), \end{aligned}$$

in which the subscripts refer to the order of the draw, A stands for "Ace" and $\overline{A}$ stands for "non-Ace." This expression is numerically evaluated as

$$P(A)=\left(\frac{4}{52}\right)\left(\frac{48}{51}\right)+\left(\frac{48}{52}\right)\left(\frac{4}{51}\right)+\left(\frac{4}{52}\right)\left(\frac{3}{51}\right)=\frac{396}{(52)(51)}=\frac{33}{221}=0.149.$$

Now calculate the probability of getting an Ace and a King in two draws.

$$P(\text{A and K}) = P(A_1)\,P(K_2|A_1) + P(K_1)\,P(A_2|K_1)$$
$$=\left(\frac{4}{52}\right)\left(\frac{4}{51}\right)+\left(\frac{4}{52}\right)\left(\frac{4}{51}\right)=\frac{32}{(52)(51)}=\frac{8}{663}=0.012.$$

An important point should be noted here. If the events "getting an Ace" and "getting a King" were independent, P(A and K) = P(A) P(K) = $(0.149)^2$ = 0.022, but it has just been shown that P(A and K) = 0.012 ≠ 0.022. Therefore, the events in question are <u>not</u> independent. At this point the reader is invited to present an argument that the events in question would have been independent if the first card drawn had been replaced in the deck and the deck had been shuffled before the second draw.

A result with important applications to fault tree analysis is the calculation of the probability of occurrence of at least one of a set of mutually independent events.

Consider the set of mutually independent events,

$$\{E_1, E_2, E_3, \ldots, E_n\}$$

and define the event $\overline{E}_1$ as the non-occurrence of E_1, the event $\overline{E}_2$ as the non-occurrence of E_2, etc. Because a particular event must either occur or not occur,

$$P(E_i)+P(\overline{E}_i)=1 \qquad \text{(B.14)}$$
$$P(\overline{E}_i)=1-P(E_i). \qquad \text{(B.15)}$$

Now there are two possibilities concerning to the events E_1, E_2, …, E_n: at least one E_i occurs or none of the E_i occur. Therefore,

$$\text{P(at least one } E_i \text{ occurs)} = \text{1-P(no } E_i \text{ occurs)}$$
$$= \text{1-P}(\overline{E}_1 \text{ and } \overline{E}_2 \ldots \text{ and } \overline{E}_n)$$

Since the E's are mutually independent, it is perhaps intuitively obvious that $\overline{E}$'s must also be mutually independent. As a matter of fact, this result can be easily proven. Therefore,

$$P(\overline{E}_1 \text{ and } \overline{E}_2 \text{ and } \overline{E}_3 \ldots \text{and } \overline{E}_n)=P(E_1)P(E_2)P(E_3)\ldots P(E_n) \qquad \text{(B.16)}$$

But from Equation B.15 this is the same as

$$[1 - P(E_1)]\,[1 - P(E_2)]\ldots[1 - P(E_n)], \tag{B.17}$$

so that the final result is

$$P(E_1 \text{ or } E_2 \text{ or } E_3 \text{ or} \ldots \text{or } E_n) = 1 - \{[1 - P(E_1)]\,[1 - P(E_2)]\,[1 - P(E_3)]\ldots[1 - P(E_n)]\}. \tag{B.18}$$

In the simple case where $P(E_l) = P(E_2) = \ldots = P(E_n) = p$, the right-hand side of Equation B.18 reduces to $1 - (1- p)^n$.

Our general result equation (Equation B.18) finds application in fault tree evaluation. For example, consider a system in which system failure occurs if any one of the events E_1, E_2, …, E_n occurs, it being assumed that these events are mutually independent. The probability of system failure is then given by Equation B.18. For example, the events E_1, …, E_n may be failures of critical components of the system, where each component failure will cause system failure. If the component failures are independent, then Equation B.18 is applicable. In the general case, the events E_1, E_2, etc., represent the modes by which system failure (top event of the fault tree) can occur. These modes are termed the minimal cut sets of the fault tree and if they are independent, i.e., no minimal cut sets have common component failures, then Equation B.18 applies.

To conclude this section, the concept of an outcome space is recalled because now the knowledge has been acquired to allow it to be defined in more detail. The elements of an outcome space possess several important characteristics:

(a) The elements of an outcome space are all *mutually exclusive*.

(b) The elements of an outcome space are *collectively exhaustive*. This means that the outcome space includes every conceivable result of the experiment.

(c) The elements of the outcome space may be characterized as being *continuous* or *discrete*; e.g., the times-to-failure for some systems are continuous events and the 52 possible cards when a card is drawn from a deck are discrete events.

B.5 Bayes' Theorem

The formula developed by Sir Thomas Bayes plays an important and interesting role in the structure of probability theory, and it is of particular significance because it illustrates a way of thinking that is characteristic of fault tree analysis. The formula is first developed using set theory, and then the meaning of the result is discussed.

Figure B-7 portrays a “partitioning” of the universal set Ω into subsets A_l, A_2, A_3, A_4, and A_5.

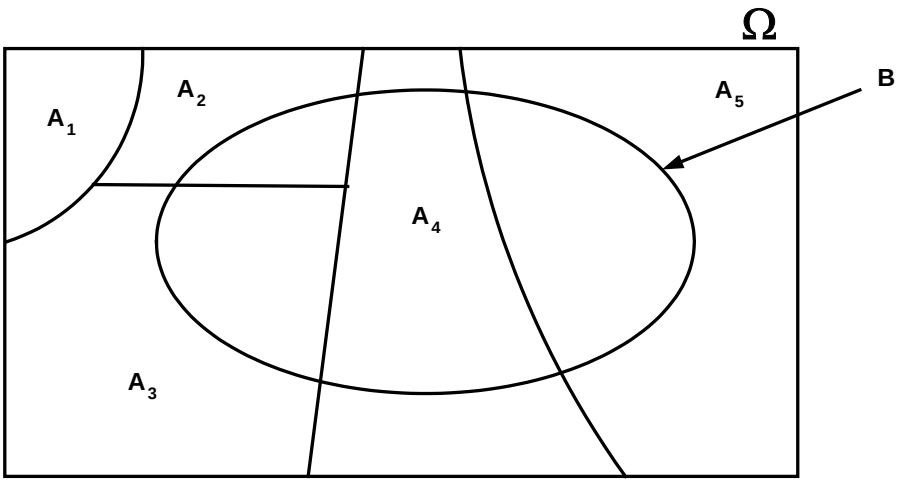


Figure B-7. Partition of the Universal Set

The A's have the following characteristics:

$$A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5 = \bigcup_{i=1}^{i=5} A_i = \Omega$$
$$A_i \cap A_j = \phi \text{ for } i \neq j. \tag{B.19}$$

Any set of A's having the properties of Equation B.19 is said to constitute a partition of the universal set. Also shown in Figure B-7 is another subset B. The reader can show (by shading the appropriate regions in the Venn diagram) that

$$(B \cap A_1) \cup (B \cap A_2) \cup (B \cap A_3) \cup (B \cap A_4) \cup (B \cap) A_5) = B.$$

(Actually $B \cap A_1 = \phi$ but $\phi \cup X = X$, where X is any set.) The expression for B above can be written in a more mathematically succinct form:

$$B = \bigcup_{i=1}^{i=5} B \cap A_i \,, \tag{B.20}$$

in which the large union symbol implies a succession of unions just as the symbol Σ implies a succession of sums. Similarly, a large intersection symbol implies a succession of intersections just as the symbol Π implies a succession of products.

Consider now the probability equation for an intersection,

$$P(A \cap B) = P(A|B)\, P(B) = P(B|A)\, P(A).$$

This is true for any arbitrary events A, B. In particular it will be true for B and any one of the A's in Figure B-7. Thus,

$$P(A_k \cap B) = P(A_k|B)\, P(B) = P(B|A_k)\, P(A_k) \tag{B.21}$$

or

$$P(A_k|B) = \frac{P(A_k \cap B)}{P(B)} = \frac{P(B|A_k)P(A_k)}{P(B)}. \tag{B.22}$$

P(B) can now be written in a different way by using Equation B.20.

$$P(B) = P\left\{\bigcup_{i=1}^{i=5} B \cap A_i\right\}$$
$$= \sum_{i=1}^{i=5} P(B \cap A_i) \;=\; \sum_{i=1}^{i=5} P(B|A_i)\, P(A_i).$$

This can be done because the events $(B \cap A_i)$ are mutually exclusive. If this expression is substituted for P(B) into Equation B.22, then:

$$P(A_k|B) = \frac{P(B|A_k)\, P(A_k)}{\sum_i P(B|A_i)\, P(A_i)}. \tag{B.23}$$

This is Bayes' theorem. The equation is valid in general for any number of events $A_1, A_2, \ldots, A_n$ that are exhaustive and are mutually exclusive (see Equation B.19). The summation then extends from i = 1 to i = n instead of i = 1 to i = 5.

Some of the interpretations of Equation B.23 will now be discussed. Suppose that some event B has been observed and that a complete list of the mutually exclusive causes of the event B can be generated. These causes are just the A's. Notice, however, the restrictions on the A's given by the relations in Equation B.19.* Now, having observed B, there may be interest in seeking the probability that B was caused by the event A_k. This is what Equation B.23 permits to be computed, if all the terms of the right-hand side can be evaluated.

The Bayesian approach is deductive—given a system event, what is the probability of one of its causative factors? This is contrasted with the inductive approach—given some particular malfunction, how will the system as a whole behave? The use of Bayes' formula will now be illustrated by a simple example for which it is particularly easy to enumerate the A's.

Consider three shipping cartons labeled I, II, III. The cartons are all alike in size, shape, and general appearance and they contain various numbers of resistors from companies X, Y, Z as shown in Figure B-8.

* The restrictions on the A_i given by Equation B.19 are equivalent to the restrictions: $\Sigma\, P(A_i) = 1$ and $P(A_i \cap A_j) = 0;\ i \neq j$.

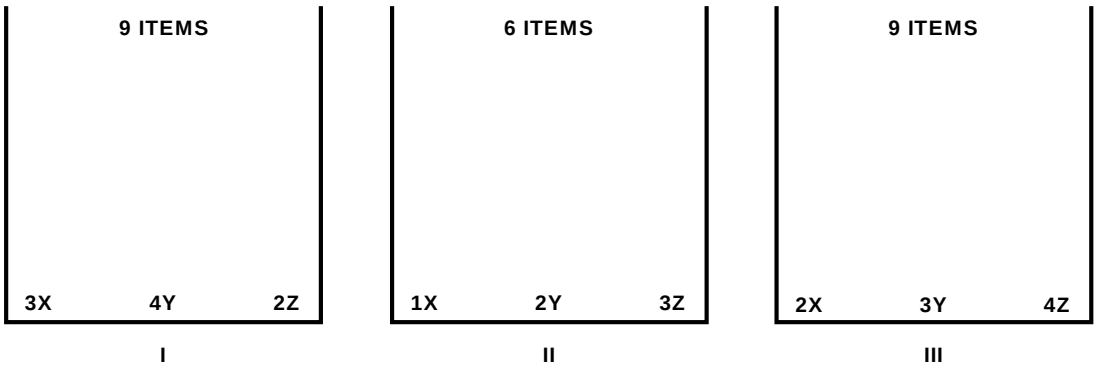


Figure B-8. Illustration of the Use of Bayes' Formula

A random experiment is conducted as follows. First, one of the cartons is chosen at random. Then two resistors are chosen from the selected box. When they are examined, it is found that both items are from Company Z. This latter event is identified with event B in this general development of Bayes' rule. The "causes" of B are readily identified: either carton I was chosen or carton II was chosen or carton III was chosen. Thus,

A_1 = choice of carton I
A_2 = choice of carton II
A_3 = choice of carton III.

Now the probability that, given event B, it was carton I that was originally chosen can be calculated.

$$P(A_1|B) = \frac{P(B|A_1)P(A_1)}{P(B|A_1)P(A_1) + P(B|A_2)P(A_2) + P(B|A_3)P(A_3)}$$

It would appear natural enough to set

$$P(A_1) = P(A_2) = P(A_3) = \frac{1}{3}$$

because the boxes are all alike and a random selection among them is made. Now the terms $P(B|A_1)$, $P(B|A_2)$ and $P(B|A_3)$ are evaluated. This is easily done from Figure B-8.

$$P(B|A_1) = \left(\frac{2}{9}\right)\left(\frac{1}{8}\right) = \frac{1}{36}$$
$$P(B|A_2) = \left(\frac{3}{6}\right)\left(\frac{2}{5}\right) = \frac{1}{5}$$
$$P(B|A_3) = \left(\frac{4}{9}\right)\left(\frac{3}{8}\right) = \frac{1}{6}.$$

Substituting these numbers into Bayes' formula:

$$P(A_1|B) = \frac{\left(\frac{1}{36}\right)\left(\frac{1}{3}\right)}{\left(\frac{1}{36}\right)\left(\frac{1}{3}\right)+\left(\frac{1}{5}\right)\left(\frac{1}{3}\right)+\left(\frac{1}{6}\right)\left(\frac{1}{3}\right)} = \frac{5}{71}.$$

In a similar way the alternative terms can be calculated

$$P(A_2|B) = \frac{36}{71}, \text{ and } P(A_3|B) = \frac{30}{71}.$$

Thus, if event B is actually observed, the chances are about 50-50 that carton II was originally chosen.

As another example, refer once more to the simple system made up of three components. It has already been determined that the system can fail in any one of the four modes, $\overline{S}_1, \overline{S}_2, \overline{S}_3, \overline{S}_4$. If the system fails and the probability that its failure mode was S_3 is of interest, this can be computed as:

$$P(\overline{S}_3|\overline{S}) = \frac{P(\overline{S}|\overline{S}_3)P(\overline{S}_3)}{P(\overline{S}|\overline{S}_1)P(\overline{S}_1)+P(\overline{S}|\overline{S}_2)P(\overline{S}_2)+P(\overline{S}|\overline{S}_3)P(\overline{S}_3)+P(\overline{S}|\overline{S}_4)P(\overline{S}_4)}.$$

This can be written in a simpler form because the system will surely fail if any one of the events $\overline{S}_1, \overline{S}_2, \overline{S}_3,$ or $\overline{S}_4$ occurs:

$$P(\overline{S}_3|\overline{S}) = \frac{P(\overline{S}_3)}{P(\overline{S}_1)+P(\overline{S}_2)+P(\overline{S}_3)+P(\overline{S}_4)}.$$

Presumably the quantities $P(\overline{S}_i)$ can be estimated from reliability data. The quantity $P(\overline{S}_i|\overline{S})$ is sometimes called the *importance* of system failure cause $(\overline{S}_i)$. Bayes' Theorem is sometimes applied to obtain optimal repair schemes and to determine the most likely contributors to the system failure (i.e., which of the $P(\overline{S}_i|\overline{S})$ is the largest).

Appendix C - Probabilistic And Statistical Analyses

This appendix presents basic probabilistic and statistical concepts that are associated with fault trees. This material augments Chapter 7, "Quantitative Evaluation of a Fault Tree." The appendix initially addresses probability distribution theory. Cumulative distribution functions and density functions as well as distribution parameters, including the mode, median, and mean are described. Measures of the dispersion of a distribution are also presented. Failure rate functions, also sometimes called hazard functions, are then defined. Examples of distributions and their parameters are given. The special case of an exponential distribution with a constant failure rate is illustrated. The appendix closes with a discussion of the concepts of Bayesian analysis, which is the current, principal statistical approach used in quantifying event probabilities.

C.1 The Cumulative Distribution Function

The symbol **X** will be used to designate the possible results of a random experiment. **X** is usually referred to as a *random variable*.* A random variable may take on values that are either discrete (e.g., the number of defective items in a lot) or continuous (e.g., the heights or weights of a population of individuals). Actually, the latter category of values is also strictly discrete because the measuring apparatus employed will have some limit of resolution. Despite this it is often convenient mathematically to consider such values as being represented by a continuous variable. In the discussion that follows, corresponding lower case letter x is used to designate a specific value of the random variable **X**.

The fundamental formula that is about to be presented will be given for the continuous case. Differences between the continuous and discrete cases will be noted when necessary. In general, the change from continuous to discrete variables is operationally a matter of replacing integral signs with summation signs. The *cumulative distribution function* F(x) is defined as the probability that the random variable **X** assumes values less than or equal to the specific value x.

$$F(x) = P\,[\mathbf{X} \leq x] \qquad \text{(C.1)}$$

According to Equation C.1, F(x) is a probability and thus must assume values only between (and including) zero and one:

$$0 \leq F(x) \leq 1.$$

If **X** ranges from $-\infty$ to $+\infty$, then

$$F(-\infty) = 0$$
$$F(+\infty) = 1.$$

If **X** has a more restricted range, $x_l < \mathbf{X} < x_u$, then

$$F(x_l) = 0$$

* Random variables will be denoted in this text by boldface type.

$$F(x_u) = 1.$$

An important property of the cumulative distribution function is that F(x) never decreases in the direction of increasing x. F(x) is a non-decreasing function although not necessarily monotonically so, in the strict mathematical sense. This can be stated more succinctly as follows:

$$\text{If } x_2 > x_1, \text{ then } F(x_2) \geq F(x_1).$$

One further important property of F(x) is stated in Equation C.2 below.*

$$P[x_1 \leq \mathbf{X} \leq x_2] = F(x_2) - F(x_1) \qquad \text{(C.2)}$$

The binomial cumulative distribution, B(x; n, p), encountered previously in this appendix is one specific example of F(x). Typical shapes for F(x) for a continuous variable and for a discrete variable are shown in Figure C-1.

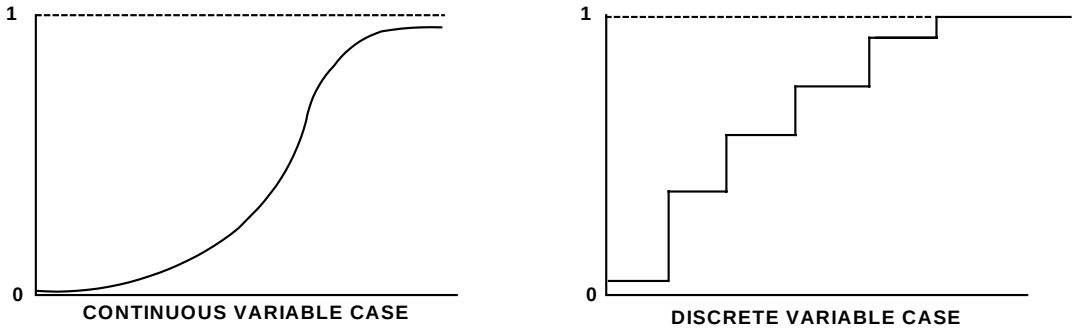
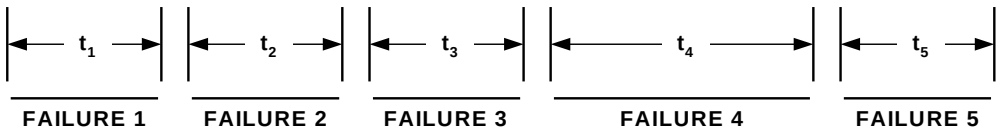


Figure C-1. Typical Shapes for F(x)

The properties of the cumulative distribution function that have been presented in the above formulae are generally valid for continuous and discrete random variables.

As an example of a random variable and its corresponding cumulative distribution, consider a random experiment in which times to failure of a single component are observed. Whenever the component fails, it is repaired, time is reset to 0 and the new time to failure is noted (see sketch below).



Assume that repair ("renewal") does not alter the component, (i.e., in every instance its repair state coincides with its initial operating state.) The random variable of interest, **T,** is the time to failure from renewal or repair. A specific value of **T** is represented by the symbol t_i. The

* For Discrete variables the formula is $P[x_1 < \mathbf{X} \leq x_2] = F(x_2) - F(x_1)$.

cumulative distribution F(t) for any t thus gives the probability that the time to failure will be less than or equal to t.

As another example consider a random experiment in which repeated measurements are performed on some item. The random variable **X** represents the "measurement outcome" in general and x_i represents some specific measurement value. The cumulative distribution F(x) gives the probability that the measurement value is less than or equal to x. Values of F(x) can be estimated from $F_{est}(x_i)$:

$$F_{est}(x_i) = \frac{n_i}{n}$$

in which n gives the total number of measurements and n_i is the number of measurements in which **X** assumes values less than or equal to x_i. As n gets larger and larger $F_{est}(x_i)$ will approach more and more closely the true value $F(x_i)$. In application, the cumulative distribution function must either be determined from theoretical considerations or be estimated by statistical methods.

C.2 The Probability Density Function

For a continuous random variable, the probability density function (pdf), f(x), is obtained from F(x) by a process of differentiation:*

$$f(x) = \frac{d}{dx} F(x). \tag{C.3}$$

An equivalent statement is

$$F(x) = \int_{-\infty}^{x} f(y)dy. \tag{C.4}$$

Because f(x) is defined as the slope of a non-decreasing function,

$$f(x) \geq 0 \tag{C.5}$$

When the pdf is integrated over the entire range of its argument, the result is unity.

$$\int_{-\infty}^{\infty} f(x)dx = 1 \tag{C.6}$$

This property of f(x) permits us to treat areas under f(x) as probabilities.

The fundamental meaning of the pdf is stated in Equation C.7.

$$f(x)dx = P[x < \mathbf{X} < x + dx] \tag{C.7}$$

* F(x) is assumed to be well-behaved and to allow differentiation.

The previous Equation C.2 can now be stated in another, especially useful, form:

$$P\left[x_1 \le \mathbf{X} \le x_2\right] = \int_{x_1}^{x_2} f(x)dx \qquad (C.8)$$

Typical shapes for f(x) are illustrated in Figure C-2 in which (a) represents a symmetrical distribution, (b) a distribution skewed to the right, and (c) a distribution skewed to the left. (In all the figures x increases to the right).

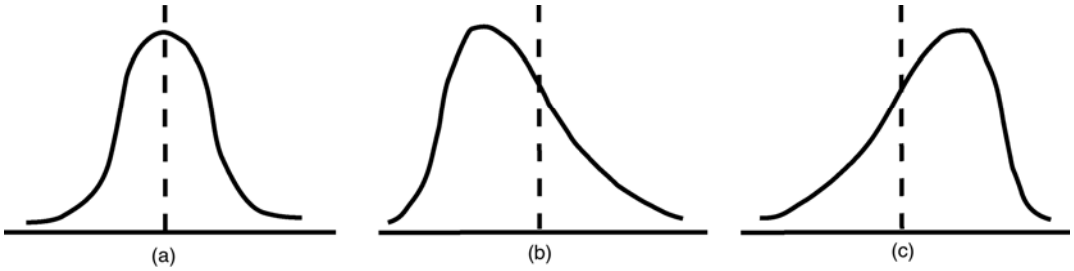


Figure C-2. Typical Shapes for f(x)

In the case of a continuous variable, probabilities must be expressed in terms of intervals. This is because the probability associated with some specific value x is always zero because there are an infinite number of values of **X** in any range. Thus f(x)dx is the probability that the quantity of interest will lie in the interval between x and x + dx. Of course, the interval length dx may be made as small as is desired. The quantity f(x) itself is therefore the probability per unit interval. In the case of discrete variables the integral signs are replaced by summation signs and the sum over the individual probabilities of the x-values in the range of interest is obtained. Equation C.8, for example, becomes applicable for a discrete variable **X** in this case.

Concerning the previous time to failure example, f(t)dt gives the probability that the component will fail in the time interval between t and t + dt. In the measurement example f(x)dx gives the probability that the measurement outcome will lie between x and x + dx. Empirically, if a large number of measurements are considered, f(x)dx could be estimated by

$$f(x)\Delta x = \frac{\Delta n_i}{n},$$

where n is the total number of measurements and Δn_i is the number of measurements for which **X** lies between x and x + Δx.

C.3 Distribution Parameters and Moments

The characteristics of particular probability density functions are described by *distribution parameters*. One of the members of this set of parameters may be used to locate the distribution along the horizontal axis. For this reason such a parameter is called a *location parameter*.

The most common location parameter is the statistical *average* or *mean*. Other location parameters commonly employed are: the *median* (50% of the area under the probability density

curve lies to the left of the median; the other 50%, to the right), the *mode*, which locates the "peak" or maximum of the probability curve (there may be no "peak" at all or there may be more than one as in bimodal or trimodal distributions); the *mid-range*, which is simply the average of the minimum and maximum values when the variable has a limited range, and others of lesser importance. For an illustration of these concepts refer to Figure C-3.

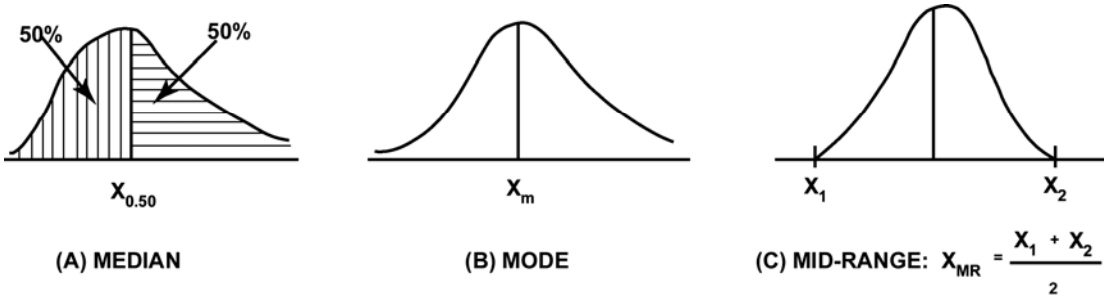


Figure C-3. The Median, Mode, and Mid-Range

In (a) of Figure C-3, the median is indicated by $x_{.50}$. From the definition of the median, 50% of the time the outcome will be less than or equal to $x_{.50}$, and 50% of the time it will be greater than $x_{.50}$. Therefore $P(\mathbf{X} \leq x_{.50}) = .50$ and, in terms of the cumulative distribution, $F(x_{.50}) = .50$. The median is a particular case of the general *α-percentile*, x_α defined such that $F(x_\alpha) = \alpha$. For example, the 90%ile is such that $F(x_{.90}) = .90$ and 90% of the time the outcome value x will be less than or equal to $x_{.90}$. In (b) of Figure C-3, the mode is indicated by x_m, and gives the most probable outcome value. In (c), the calculation of the mid-range from the two extreme values x_1 and x_2 is illustrated.

The average is also termed the mean, or the expected value. If a random experiment is repeated many times and the outcome values are averaged, this empirical average will approximate the true average and will approach the true average more and more closely as the number of repetitions is increased. (This assumes that the distribution has an average and is such that the empirical average converges to the population average.)

In the case of uni-modal symmetrical distribution as shown in (a) of Figure C-2, the mean, median, and mode all coincide. For skewed distributions, as in Figure C-2 (c), the median will fall between the mode and the mean.

In Figure C-4 two symmetrical distributions with the same values of mean, median, and mode are displayed. They are, however, strikingly different from the perspective of the clustering about the central position. Parameters used to describe this aspect of a distribution are known as *dispersion parameters*. Of these, the most familiar are the variance and the square root of the variance, or *standard deviation*. Other dispersion parameters, less frequently employed, are the median absolute deviation and the range between a lower bound value and an upper bound value. The variance of a distribution will be discussed later.

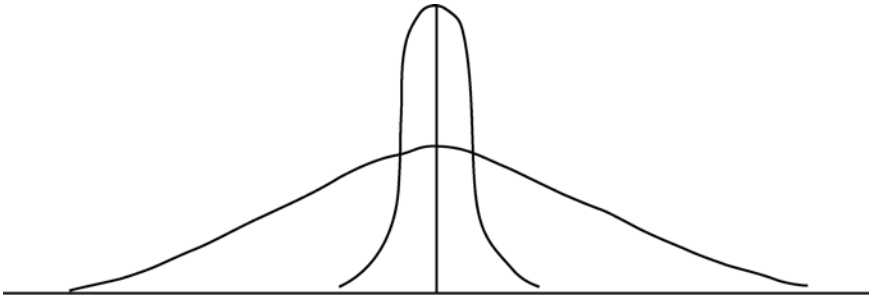


Figure C-4. Two Symmetrical Distributions with Different Dispersion Parameters

There are, in fact, many other types of distribution parameters, but those that have been mentioned are some of the most basic ones. It is essential that the general methods for calculating distribution parameters, once the functional form of the pdf is known, become familiar to the analyst. Some of these general methods entail calculating the *moments of the distribution* and are of great importance in theoretical statistics. Distribution moments may be calculated about any specified point but here the calculation is restricted to the discussion of (a) moments about the origin, and (b) moments about the mean.

Moments About the Origin

The first moment about the origin is defined as

$$\mu_1' = \int_{-\infty}^{\infty} x\ f(x)dx. \tag{C.9}$$

This defines the mean or expected value of **X,** written E[**X**]**.** The symbol μ will be used for the mean for the sake of brevity although actually $E[\mathbf{X}] = \mu_1'$.

The second moment about the origin is defined as

$$\mu_2' = \int_{-\infty}^{\infty} x^2 f(x)dx, \tag{C.10}$$

and yields the expected value of $\mathbf{X}^2$, E[$\mathbf{X}^2$].

In general, the n^{th} moment about the origin is defined as

$$\mu_n' = \int_{-\infty}^{\infty} x^n f(x)dx, \tag{C.11}$$

and yields the expected value of $\mathbf{X}^n$, E[$\mathbf{X}^n$].

If **Y** = g(**X**) is any function of **X**, and **X** is distributed according to the pdf f(x), then the expected value of g(**X**) may be obtained from

$$E[Y] = E[g(X)] = \int_{-\infty}^{\infty} g(x)f(x)dx. \tag{C.12}$$

Moments About the Mean

The first moment about the mean is defined as

$$\mu_1 = \int_{-\infty}^{\infty} (x - \mu) f(x) dx. \tag{C.13}$$

Because, μ_1 is always and invariably equal to 0, it is not of great utility.

The second moment about the mean is defined as

$$\mu_2 = \int_{-\infty}^{\infty} (x - \mu)^2 f(x) dx. \tag{C.14}$$

This gives the variance σ^2 or $E[(\mathbf{X}-\mu)^2]$. In general, the n^{th} moment about the mean is defined as

$$\mu_n = \int_{-\infty}^{\infty} (x - \mu)^n f(x) dx, \tag{C.15}$$

and yields $E[(\mathbf{X}-\mu)^n]$.

There is a useful relationship between μ_2, μ_2', and μ_1' namely,

$$\mu_2 = \mu_2' - (\mu_1')^2 \tag{C.16}$$

Equation C.16 permits calculation of the variance by evaluating the integral in Equation C.10 rather than the integral in Equation C.14, which is more complicated algebraically. Equation C.16 is easily proven as follows:

$$\begin{aligned} \mu_2 &= \int_{-\infty}^{\infty} (x-\mu)^2 f(x) dx \\ &= \int_{-\infty}^{\infty} x^2 f(x) dx - 2\mu \int_{-\infty}^{\infty} x f(x) dx + \mu^2 \int_{-\infty}^{\infty} f(x) dx \\ &= \mu_2' - 2\mu^2 + \mu^2 = \mu_2' - \mu^2 = \mu_2' - (\mu_1')^2. \end{aligned}$$

In the case of a <u>discrete</u> random variable, the first moment about the origin is written

$$\mu = \mu_1' = \sum_{i=1}^{n} x_i \, p(x_i), \tag{C.17}$$

where $p(x_i)$ is the probability associated with the value x_i. The commonly used formula for finding the average of n values,

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i,$$

is a special case of Equation C.17 which can, be used whenever all the individual values are treated as having the same probability or "weight," namely 1/n. Thus, in the case of a single die

$$\mu = \mu' = \frac{1+2+3+4+5+6}{6} = \frac{21}{6} = 3.5,$$

so that the expected value is 3.5 despite the fact that such an outcome is impossible in practice.

Also, if the random variable is discrete, the second moment about the mean (the variance) assumes the form

$$\mu^2 = \sum_{i=1}^{n} (x_i - \mu)^2 \, p(x_i). \tag{C.18}$$

If all the x_i have the same "weight" $\frac{1}{n}$, Equation C.18 reduces to a sampling formula for computing the variance of a sample of n readings,*

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 . \tag{C.19}$$

This section is concluded with a simple example of the use of distribution moments. Consider the rectangular pdf shown in Figure C-5 in which any value between a and b is equally likely. Because any value is equally likely, f(x) = f_0, a constant. The area under a pdf must integrate to 1 and hence:

$$\text{Area} = f_0 (b - a) = 1 \text{ so that } f_0 = \frac{1}{b - a}.$$

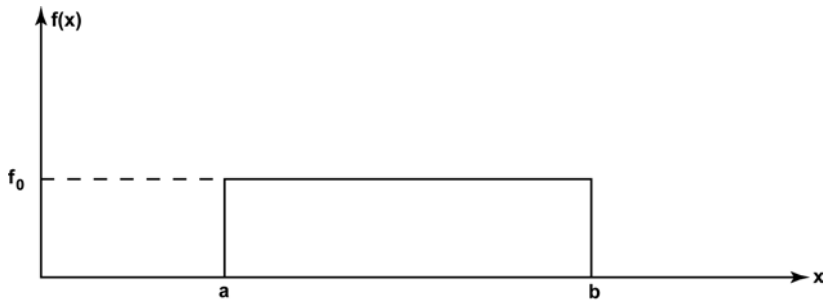


Figure C-5. The Rectangular Distribution

The mean of this distribution is given by

* Equation C.18 provides a biased estimate of the population variance α^2. The bias is eliminated by multiplying Equation C.18 by the factor n/(n-1), "Bessel's correction." For large n the difference is slight.

$$\mu = \mu_1' = E[X] = \int_a^b x \frac{1}{b-a}\, dx$$
$$= \frac{1}{b-a}\left[\frac{x^2}{2}\right]_a^b = \frac{b^2-a^2}{2(b-a)} = \frac{(b-a)(b+a)}{2(b-a)} = \frac{b+a}{2}.$$

The variance of the distribution is given by

$$Var = \alpha^2 = \mu_2 = \mu_2' - (\mu_1')^2 = \int_a^b x^2 \left(\frac{1}{b-a}\right) dx - \left(\frac{b+a}{2}\right)^2$$
$$= \left(\frac{1}{b-a}\right)\left[\frac{x^3}{3}\right]_a^b - \left(\frac{b+a}{2}\right)^2$$
$$= \left(\frac{1}{b-a}\right)\left(\frac{b^3-a^3}{3}\right) - \left(\frac{b+a}{2}\right)^2$$
$$= \left(\frac{1}{b-a}\right)\left[\frac{(b-a)(b^2+ab+a^2)}{3}\right] - \left(\frac{b+a}{2}\right)^2$$
$$= \frac{b^2-2ab+a^2}{12} = \frac{(b-a)^2}{12}.$$

C.4 The Failure Rate or Hazard Function

Recall from previous sections that

$$F(t) = P[\text{failure occurs at some time prior to } t]$$

and that

$$f(t)dt = P[\text{failure occurs between } t \text{ and } t + dt].$$

A conditional probability, $\lambda(t)$, called the failure rate function or hazard function can now be defined:

$$\lambda(t)dt = P[\text{failure occurs between } t \text{ and } t + dt \mid \text{no prior failure}]. \quad (C.20)$$

For any general distribution, there is an important relationship between the three functions $\lambda(t)$, $f(t)$, and $F(t)$;

$$\lambda(t) = \frac{f(t)}{1-F(t)}. \quad (C.21)$$

The validity of Equation C.21 is easily demonstrated as follows.

Let the time at which failure occurs be designated as **T**. Since **T** is a random variable, according to the definition given in Equation C.20,

$$\lambda(t)dt = P\,[t<\mathbf{T}<t+dt \mid t<\mathbf{T}].$$

Let (t<**T**<t+dt) be denoted as event A and (t < **T)** be denoted as event B. Remembering that in general,

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Therefore,

$$\lambda(t)dt = \frac{P[(t < \mathbf{T} < t + dt) \cap (t < \mathbf{T})]}{P(t < \mathbf{T})}.$$

Now event A is just a special case of event B, i.e., when A occurs then B automatically occurs. Since A is a subset of B, A∩B=A. It follows that

$$\lambda(t)dt = \frac{P[t < \mathbf{T} < t + dt]}{P(t < \mathbf{T})} = \frac{P[A]}{P[B]} = \frac{f(t)dt}{1 - F(t)}.$$

Finally,

$$\lambda(t) = \frac{f(t)}{1 - F(t)},$$

which is just Equation C.21.

If λ(t) is plotted against time for a general system, the curve shown in Figure C-6 results. For obvious reasons this relationship between λ(t) and t has become known as the "bathtub curve."

Figure C-6. Plot of λ(t) vs. t for a General System

This curve can be divided into three parts that are labeled I, II, III in Figure C-6. Region I is termed the region of "infant mortality" where sometimes an underlying distribution is difficult to determine. The distribution appropriate to this part of the curve may depend quite critically on

the nature of the system itself. Manufacturers will frequently subject their product to a burn-in period attempting to eliminate the early failures before lots are shipped to the consumer. Region II corresponds to "a constant failure rate function," and is the region of chance failures to which the exponential distribution applies. Region III corresponds to a wear-out process for which the normal distribution or the more general Weibull Distribution often provide adequate models. For an actual system, the λ(t)-vs.-t curve may be quite different from that depicted in Figure C-6. For example, the exponential Region II may be entirely missing or the burn-in region may be negligible. Figure C-7 shows the λ(t) of the Space Shuttle main engine (SSME) and the solid rockets (ISRB) vs. burntime. Notice that the SSME has a flat hazard throughout its burn at least until the ISRBs are separated (until SRB sep). The ISRBs on the other hand follow a more traditional "bathtub" curve relationship.

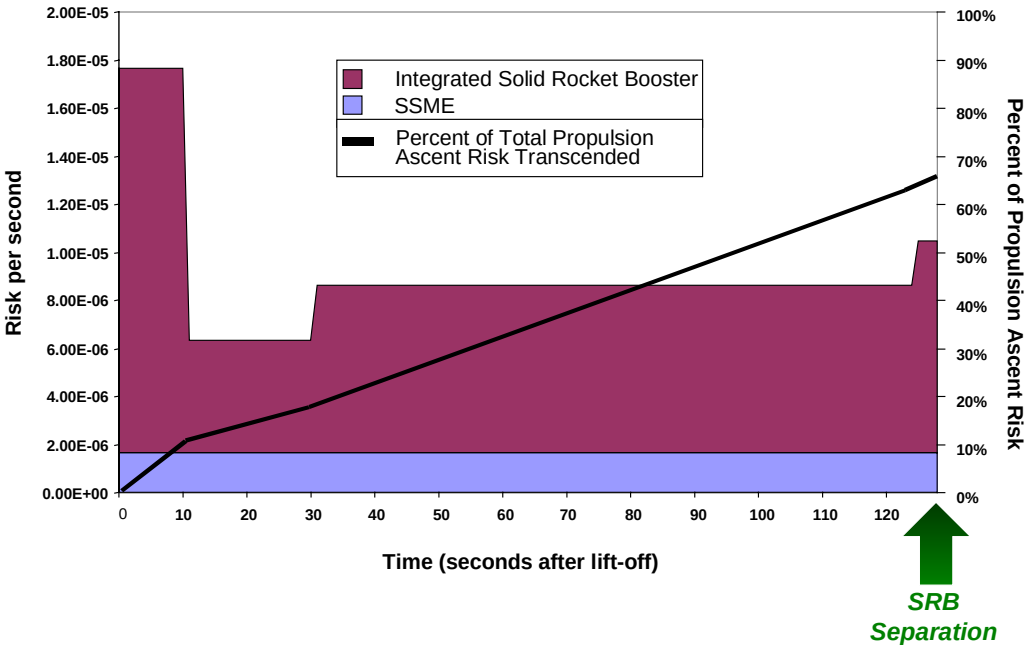


Figure C-7. Risk Intensity vs. Time During Shuttle Ascent

Returning to the failure rate function equations, it is convenient to solve Equation C.21 explicitly for both F(t) and f(t). This is accomplished by writing Equation C.21 in the form

$$\lambda(t)dt = -\frac{[-F'(t)dt]}{1-F(t)}, \tag{C.22}$$

where

$$F'(t) = -\frac{dF(t)}{dt}.$$

Integrating both sides of Equation C.22,

$$-\int_0^t \lambda(x)dx = 1n\left[1-F(t)\right].$$

This is equivalent to

$$1 - F(t) = \exp\left[-\int_0^t \lambda(x)dx\right].$$

and therefore

$$F(t) = 1 - \exp\left[-\int_0^t \lambda(x)dx\right]. \quad \text{(C.23)}$$

If Equation C.36 is differentiated,

$$f(t) = \lambda(t)\exp\left[-\int_0^t \lambda(x)dx\right]. \quad \text{(C.24)}$$

If $\lambda(t) = \lambda$ (a constant) in Equation C.23 and C.24, the reader should be able to show that

$$F(t) = 1 - e^{-\lambda t} \text{ and } f(t) = \lambda e^{-\lambda t},$$

which is the exponential distribution. For the exponential distribution, then, the failure rate function is a constant (independent of t) and

$$\lambda(t)dt = P\,[\text{failure occurs between t and t + dt} \mid \text{no prior failure}] = \lambda dt.$$

If a component failure distribution is described using the exponential distribution, then this implies the constant, steady state portion of the bathtub curve with no burn-in or wear-out occurring. Because of the constancy of the failure rate function in this case, the exponential distribution is often referred to as the "random failure distribution," i.e., the probability of future failure is independent of previous successful operating time.

It is also valuable to note that if $e^{-t/\theta}$ is used to represent reliability, this is a conservative representation even if wear-out is occurring (but no burn-in), i.e., $R(t) \geq e^{-t/\theta}$ where R(t) is the actual reliability and θ is the actual mean time to failure. This relation is true for $t \leq \theta$ [1].

Equations C.23 and C.24 may be used to investigate a wide variety of failure rate models. For instance, an assumption that $\lambda(t) = kt$ (linearly increasing failure rate) implies that

$$R(t) = 1 - F(t) = \exp(-kt^2/2),$$

which is known as the *Rayleigh distribution*. An important distribution of times-to-failure, the *Weibull distribution*, is obtained by putting $\lambda(t) = Kt^m$ $(m > -1)$, whence

$$f(t) = kt^m \exp\left(-\frac{kt^{m+1}}{m+1}\right)$$

and

$$R(t) = 1 - F(t) = \exp\left(-\frac{kt^{m+1}}{m+1}\right).$$

The Weibull distribution is a two-parameter distribution, where k is known as the *scale parameter* and m as the *shape parameter*. For m = 0 the Weibull becomes equivalent to the exponential distribution and as m increases a wear-out behavior is modeled. When m increases to 2, f(t) approaches normality. When m is less than 0 but greater than - 1, the burn-in portion of certain bathtub curves may be modeled. Thus, by changing the value of m, the Weibull distribution can be used to represent the hazard function in regions I, II, or III of the bathtub curve. The reader can find a more detailed description of the Weibull distribution in the reliability literature [2] pp. 137-138, [3], and [4], p. 190 et seq.

C.5 Bayesian Analyses

When a Bayesian approach is applied, the parameters of the distribution are treated as random variables and are not fixed. Considering the exponential distribution $f(x) = \frac{1}{\theta}e^{-x/\theta}$, the mean time to failure θ can be treated as being associated with a probability distribution. Expressing the exponential in terms of the failure rate λ = 1/θ, $f(x) = \lambda e^{-\lambda x}$. The failure rate is thus also associated with a probability distribution (because of the relation λ = 1/θ, the distribution of λ is given by the distribution for θ and vice versa). From here on $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$ will denote the random variables associated with the parameters λ and θ.

Let the pdf for $\boldsymbol{\lambda}$ be denoted by p(λ). The pdf p(λ) is known as the *prior distribution*. It represents the prior knowledge of $\boldsymbol{\lambda}$ before a given sample is taken. Assume now that a given sample $(t_1, t_2, \ldots, t_n)$ of component failure times is collected. In this case the *posterior distribution* of $\boldsymbol{\lambda}$ can be developed that represents the updated knowledge of the distribution of $\boldsymbol{\lambda}$ with the additional sample data incorporated.

The posterior distribution of $\boldsymbol{\lambda}$, whose pdf is denoted by p(λ|D), is easily obtained by applying Bayes theorem* (the symbol "D" denotes the data sample, e.g., $(t_1, t_2, \ldots, t_n)$). Bayes theorem states

$$P(B|A) = \frac{P(A|B)\,P(B)}{\sum P(A|B)P(B)}.$$

Letting "A" denote the data sample "D," and "B" the event that the failure rate lies between λ and λ + dλ,

* See Appendix B, Section B.5.

$$p(\lambda|D) = \frac{\exp\left[-\sum_{i=1}^{n} \lambda t_i\right] \lambda^n p(\lambda)}{\int \exp\left[-\sum_{i=1}^{n} \lambda t_i\right] \lambda^n p(\lambda)\, d\lambda}.$$

Here the summation sign is replaced by the integral sign. Because the denominator does not involve λ (it is integrated out) the above equation can be written as

$$p(\lambda \mid D) = K \exp\left[-\sum_{i=1}^{n} \lambda t_i\right] \lambda^n p(\lambda),$$

where K is treated as a normalizing constant. The distribution p(λ|D) is the posterior distribution of λ now incorporating both the prior knowledge and the observed data sample.

Bayes theorem thus gives a formal way of updating information about the failure rate λ (i.e., going from p(λ) to p(λ|D)). If a second sample D' were collected (say $t_1', t_2', \ldots, t_n'$) then the distribution of λ could be updated to incorporate both sets of data. If p(λ|D,D') represents the posterior distribution of λ based on both sets of data D and D' then the above equation is simply used with p(λ|D) now as the prior giving

$$p(\lambda|D, D') = K \exp\left[-\sum_{i=1}^{n} \lambda t_i\right] \lambda^n p(\lambda|D).$$

Choices for the initial prior p(λ) as well as techniques for handling various kinds of data are described in detail in various texts [5], [6]. In Bayesian approaches, the pdf's obtained for the parameters (such as p(λ|D)) give detailed information about the possible variability and uncertainty in these parameters. Point values, such as the most likely value of λ or the mean value of λ can be obtained. Interval values, which are probability intervals and are sometimes called Bayesian confidence intervals can also be obtained. For example, having determined p(λ|D) the lower and upper 95% values of λ_L and λ_U can be determined, such that there is a 95% probability that the failure rate lies between these values, i.e.,

$$\int_{\lambda_L}^{\lambda_U} p(\lambda|D) d\lambda = 0.95.$$

Other bounds and other point values can be obtained in the Bayesian approach because the parameter distribution (e.g., p(λ|D)) is entirely known and this distribution represents the state of knowledge about the parameter. The Bayesian approach has the advantage that engineering experience and general knowledge, as well as "pure" statistical data, can be factored into the prior distribution (and hence posterior distribution). Once distributions are obtained for each relevant component parameter, such as the component failure rate, then the distribution is straightforwardly obtained for any system parameter quantified in the fault tree, such as the system unavailability, reliability, or mean time to failure. The priors that truly represent the

analyst's knowledge must be determined very carefully, including ascertaining the impact of different priors if they are all potentially applicable [6].

C.6 References

1. B. Gnedenko, Y Belyayev, and A. Solovyev, *Mathematical Methods of Reliability Theory*, Translation edited by R.E. Barlow, Academic Press, New York 1969.

2. D. Lloyd, M. Lipow, *Reliability: Management, Methods, and Mathematics*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1962.

3. H. Romig, *Binomial Tables*, John Wiley and Sons, Inc., New York, 1953.

4. M. Shooman, *Probabilistic Reliability: An Engineering Approach*, McGraw-Hill, New York, 1968.

5. C. Rao, *Linear Statistical Inference and Its Applications*, John Wiley and Sons, Inc., New York, 1977.

6. N. Mann, R. Schafer and N. Sigpurwalla, *Mathematical Methods for Statistical Analysis of Reliability and Life Data*, John Wiley and Sons, inc., New York, 1974.

Appendix D. Markov Modeling for Reliability Analysis

This appendix provides a brief introduction to the use of continuous time Markov models for reliability analysis of non-repairable systems. Specifically, the Markov model as used for the solution of a dynamic fault tree is discussed

A dynamic fault tree (DFT)* model cannot be easily evaluated using traditional fault tree analysis techniques, such as those that are based on cut sets or other Boolean logic techniques. Since the DFT gates must capture the order in which events occur, not simply their probability of occurrence, a Markov model is used for solution. The equivalent Markov model can be generated from the DFT and then solved using ordinary differential equations.

D.1 Introduction to Markov Modeling

As a simple example of a sequence dependent failure, consider a system with one active component and one standby spare connected with a switch controller shown in Figure D-1. (This example system was considered earlier in Chapter 8). Assume that the failure mode of the switch is such that when it fails, it is unable to switch. The failure of the switch only matters if a switch from the primary to the standby spare is required. If the switch controller fails *after* the primary unit fails (and thus the standby spare is already in use), then the system can continue operation. However, if the switch controller fails *before* the primary unit fails, then the standby spare unit cannot be switched into active operation and the system fails when the primary unit fails. The order in which the primary and switch fail determines whether the system continues operation.

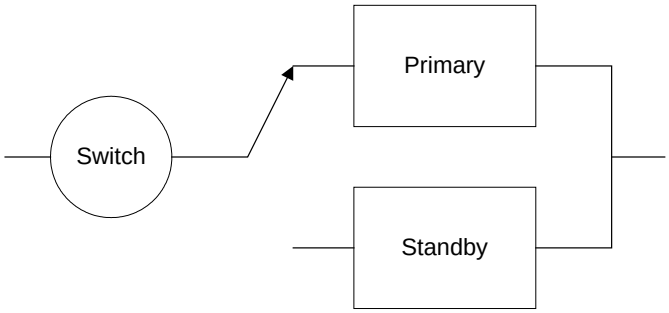


Figure D-1. Example of Sequence Dependent Failure

The DFT for this standby system is shown in Figure D-2, where a Priority-AND gate is used to model the order dependence.

* Dynamic Fault Trees are described in Chapter 8.

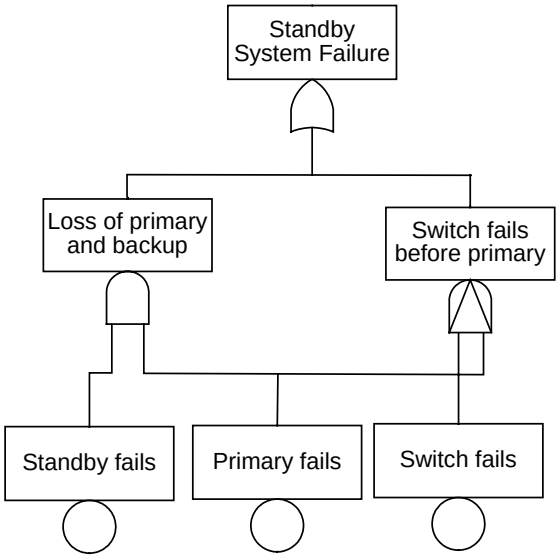


Figure D-2. DFT for Standby Space System

Figure D-3 shows the Markov model for the standby spare system, where circles represent states and arcs represent events that cause transitions between states. Arcs are labeled with the rate at which the transition occurs (usually the failure rate of the failed component). The failure rate of an active component (the primary or spare when activated) is λ_p, while λ_s is the failure rate of the switch. In the initial state (state 1) the primary, spare and switch are all functional. The two transitions from state 1 represent the failure of the primary (leading to state 2) and the failure of the switch (leading to state 3). In state 2, the spare has been switched to active service, and a later failure of the switch is inconsequential. From state 3, failure of the primary leads to system failure (state F).

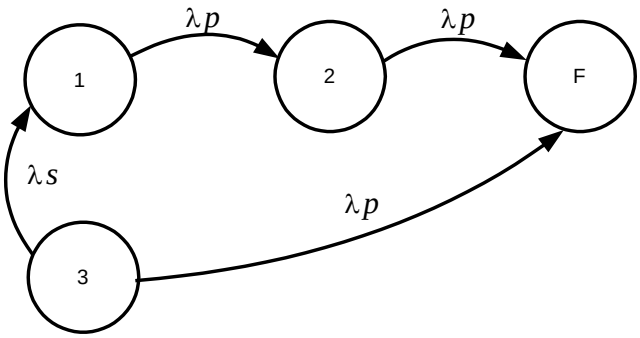


Figure D-3. Markov Model for Standby Spare System.

A Markov chain is equivalent to a set of differential equations, with one equation for each state. Associated with each state is a state variable representing the time-dependent probability that the system is in that state (note that the states are mutually exclusive). The set of differential equations associated with the Markov chain shown in Figure D-3 is given by

$$\frac{d}{dt}P_1(t) = -(\lambda_p + \lambda_s)P_1(t)$$
$$\frac{d}{dt}P_2(t) = \lambda_p P_1(t) - \lambda_p P_2(t)$$
$$\frac{d}{dt}P_3(t) = \lambda_s P_1(t) - \lambda_p P_3(t)$$
$$\frac{d}{dt}P_F(t) = \lambda_p P_2(t) + \lambda_p P_3(t).$$

where it is assumed that the initial state (at time zero) is state 1:

$$P_1(0) = 1 \text{ and } P_i(t) = 0, i \neq 1$$

The solution of the differential equations is given by

$$P_1(t) = e^{-(\lambda_p+\lambda_S)t}$$
$$P_2(t) = \frac{\lambda_p}{\lambda_S}(e^{-\lambda_p t} - e^{-(\lambda_p+\lambda_S)t})$$
$$P_3(t) = e^{-\lambda_p t} - e^{-(\lambda_p+\lambda_S)t}$$
$$P_4(t) = 1 - \frac{\lambda_p + \lambda_S}{\lambda_S}e^{-\lambda_p t} + \frac{\lambda_p}{\lambda_S}e^{-(\lambda_p+\lambda_S)t}.$$

The reliability of the standby system is the probability that the system is not in a failed state:

$$R(t) = \frac{\lambda_p + \lambda_S}{\lambda_S}e^{-\lambda_p t} - \frac{\lambda_p}{\lambda_S}e^{-(\lambda_p+\lambda_S)t}$$

Additional detail on Markov models for reliability is provided in [1].

D.2 Converting a Dynamic Fault Tree to a Markov Model

The algorithm for converting a dynamic fault tree to an equivalent Markov model is conceptually simple [2]. Starting with the initial state (all components operational), a set of target states and associated transitions are generated by considering the effect of failing each active component, one at a time. Each time a new state is generated, it is checked against the fault tree to see if it is an operational or failed state. Each operational target state is added to the Markov chain and is further expanded.

Consider an example static fault tree, consisting of three processors and two memories connected by a bus. At least one processor, one memory and the bus are needed for the system to operate. The fault tree model for this 3P2M (three processor and two memory) system is shown in Figure D-4, and the corresponding Markov chain is shown in Figure D-5. In Figure D-5, λ is the failure rate of a single processor; μ is the failure rate of a single memory and σ is the failure rate of the bus. Each state is labeled with three integers, representing the number of operational

processors, memories and buses, respectively. The initial state (3,2,1) has three outgoing transitions, representing a failure of a single component. Note that the transition rates are multiplied by the number of active components of each type. The failure states (FP, FM and FB), representing failures caused by exhaustion of processors, memories and buses, respectively, are drawn separately to simplify the diagram. Although it is simpler and more efficient to solve the 3P2M example system directly as a fault tree, the Markov chain solution yields identical results.

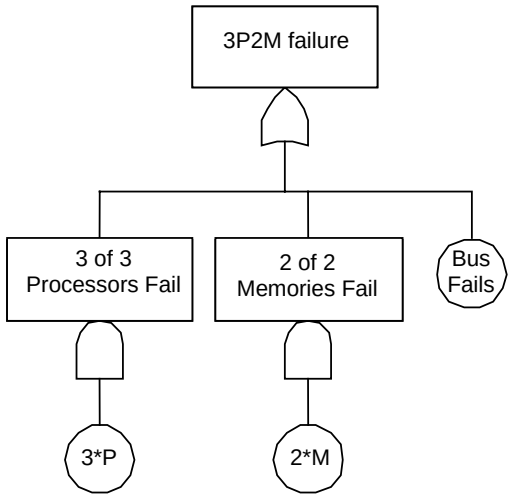


Figure D-4. 3P2M Example System

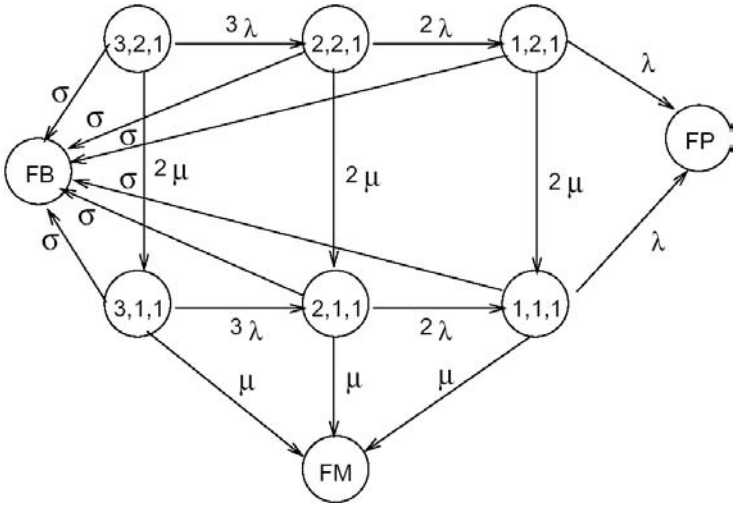


Figure D-5. Markov chain for 3P2M system

Because this 3P2M system has no order dependent behavior (since it only needs AND and OR gates in the fault tree), separate paths can lead to the same state. For example, consider state (2,1,1), in which there has been both a processor failure and a memory failure. The two paths from the initial state (3,2,1) to the (2,1,1) state represent the different orders in which the two failures could have occurred. The single state (2,1,1) can be used as the target state because the order is not important for the analysis of the fault tree. However, if there had been a gate in the

fault tree that depended on the order, then the state (2,1,1) would have been split into two states, recording the order of the events.

The Markov chain approach is necessary when there are dynamic gates in a fault tree. The Markov chain generation algorithm identifies which sets of events are order-dependent and generates separate states for each possible order. When spare gates are used (warm spare gates (WSP) in this example), the assignment of the spares as well as the determination of the failure rate for each spare (active or not) is included. Functional dependencies affect the set of operational components in the target state, and can interact with spare gates. Thus, although the Markov chain generation is conceptually simple, there are interaction and timing issues to be considered. As a slightly more involved example, consider a DFT for a Vehicle Management System (VMS) (Figure D-6) and its associated Markov chain (Figure D-7), where transitions are labeled with the basic event that occurs rather than with the transition rate, for clarity. Notice in Figure D-7 that there are two states representing the combined failure of components A and C. State S_7, representing the sequence *A fails* then *C fails* is a failure state. This is a failure state because of the asymmetry of the system: if A fails it is replaced by the spare. When C fails, the spare is not available and the system fails. Contrast state S_7 with state S_{13}, representing the sequence *C fails* then *A fails*. State S_{13} is an operational state for this system.

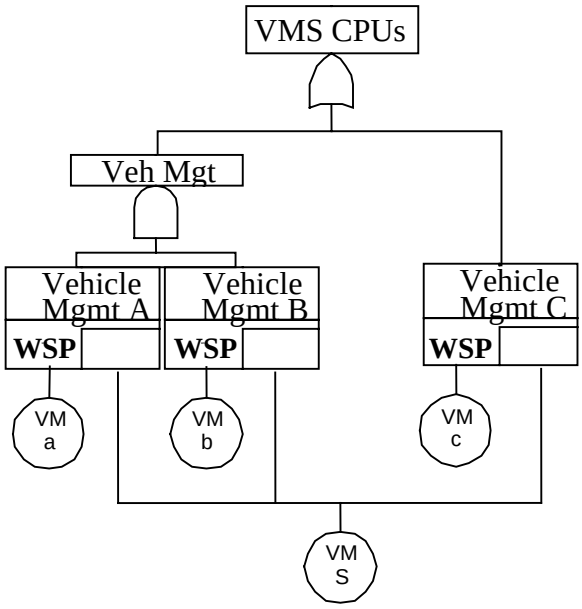


Figure D-6. An Example DFT for VMS

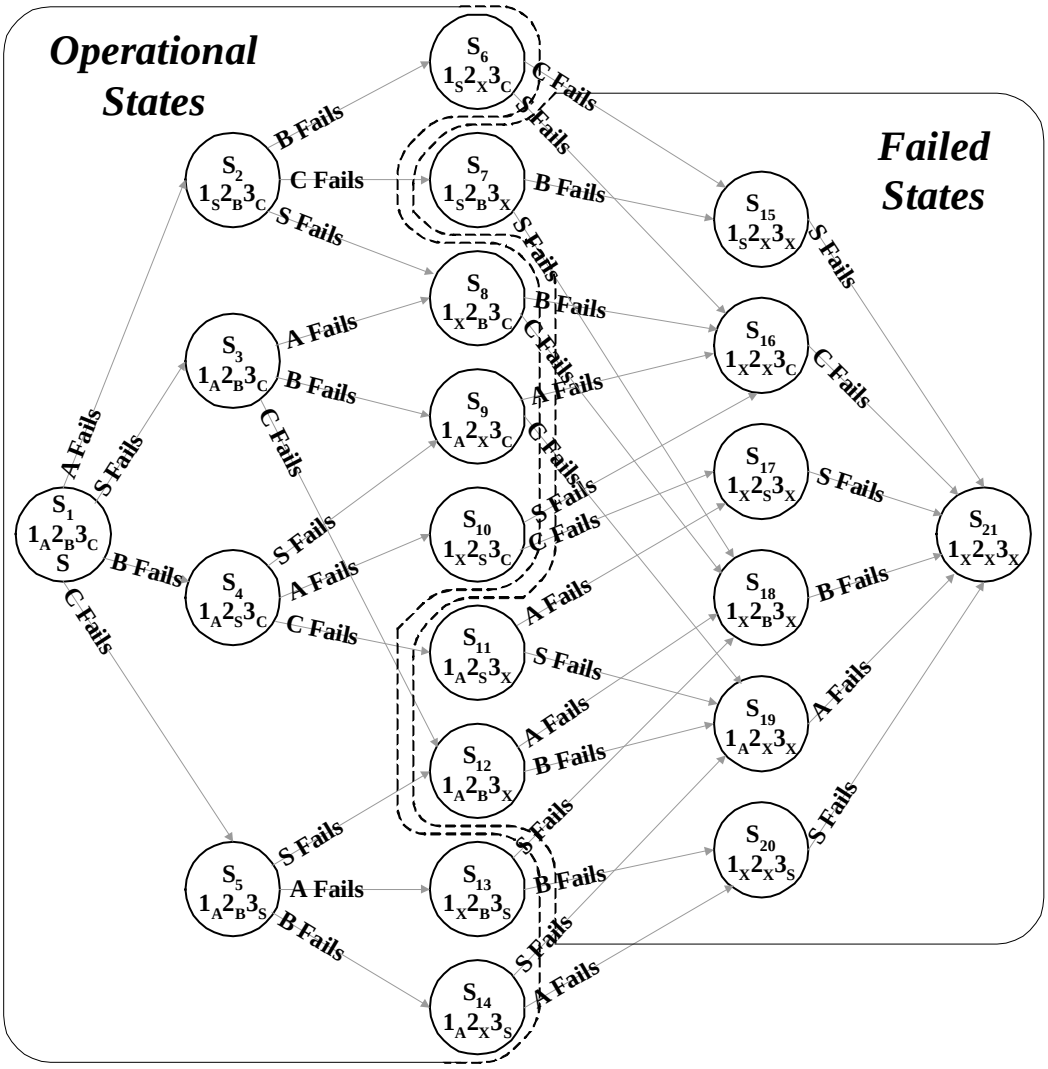


Figure D-7. Markov Model for VMS

As an illustration of the effect of the FDEP (functional dependency) gate, consider the memory system from the HECS example described in Chapter 13 (shown in Figure D-8). The main part of the HECS memory system is a set of redundant memory units (M1 through M5) of which three are required. Thus when three of the five fail, the memory system fails. However, the memory units are functionally dependent on the memory interface units (MIUs). If MUI 1 fails, it disconnects M1 and M2. If MIU 2 fails, it disconnects M3 and M4. If both MIUs fail, then M3 is disconnected.

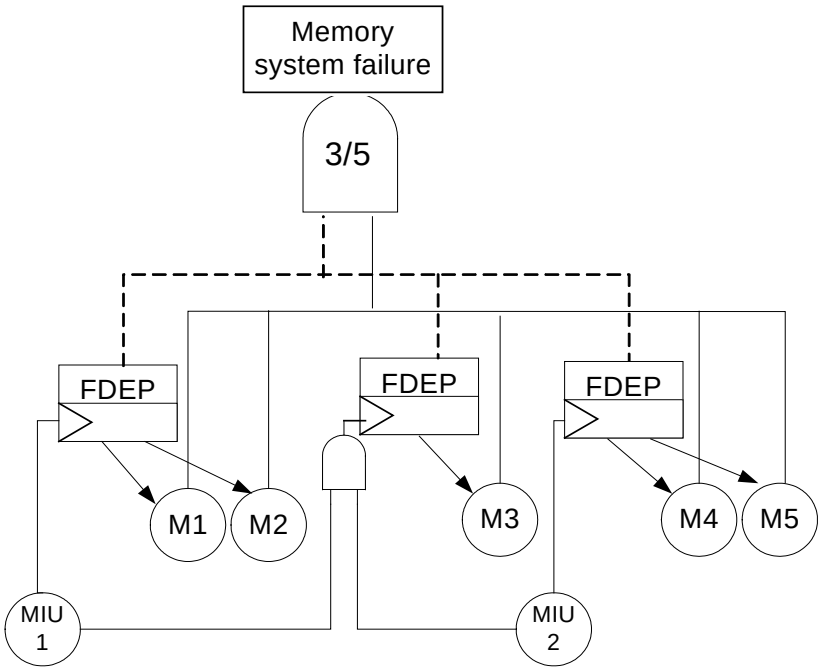


Figure D-8. DFT for HECS Memory System

The Markov chain for the HECS memory system is shown in Figure D-9. Figure D-9 shows more than the enumeration of the states associated with up to two memory units failed. Each individual memory unit failure moves the system from a state with i failures to a state with i+1 failures. Notice that some arcs (those representing MIU failures) take the system from a state where there are i failures to i+3 failures (the MIU as well as its dependent memory units).

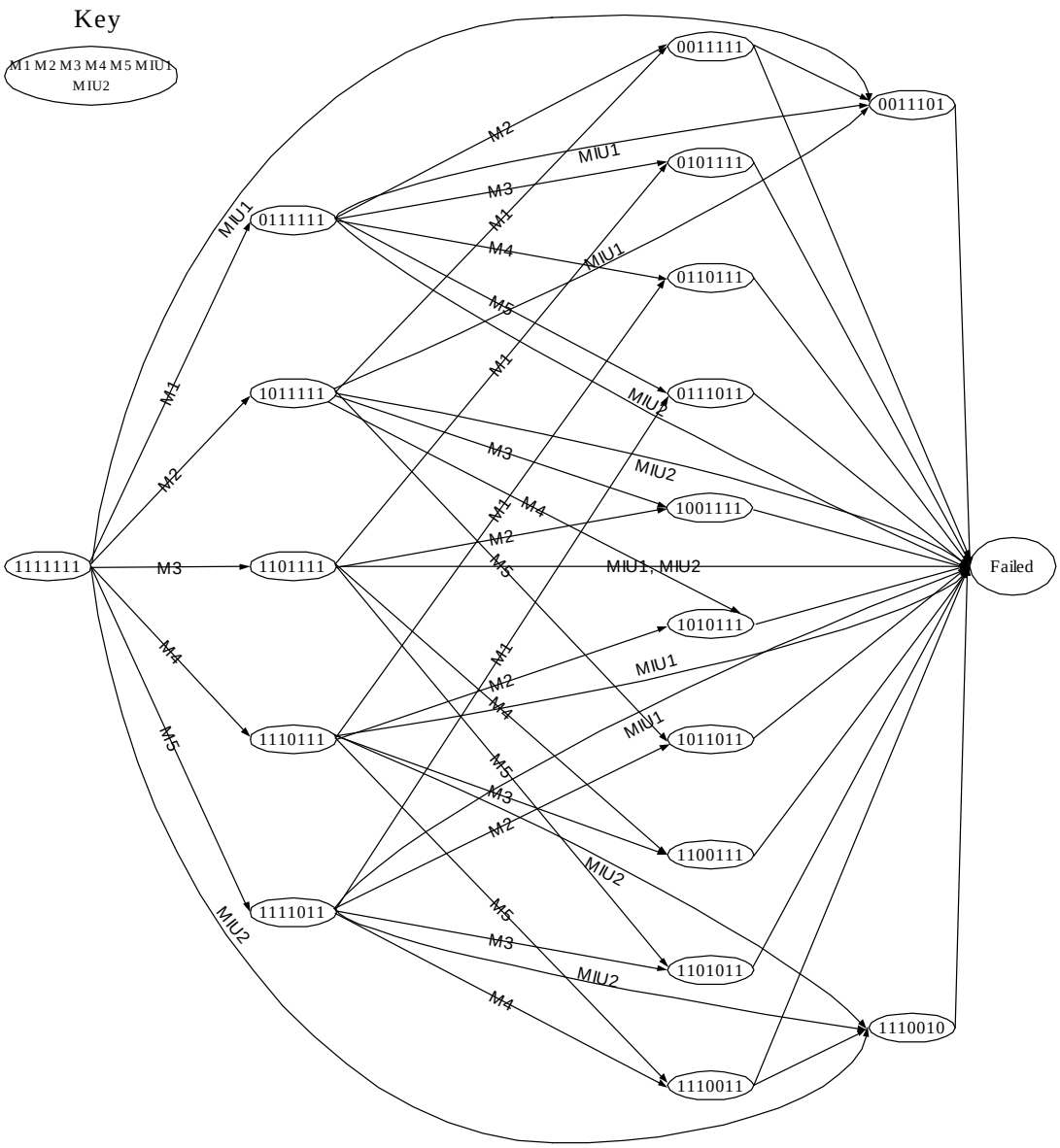


Figure D-9. Markov Model for HECS Memory System

Once the Markov chain is generated, its solution is fairly straightforward using numerical techniques. However, the Markov chain can be extremely large; the number of states is exponential in the number of components. Fortunately, good techniques exist for truncating the Markov chain and generating an approximate (bounded) result.

D.3 References

1. J. Pukite and P. Pukite, "Modeling for Reliability Analysis," *IEEE Press*, 1998.

2. J. Dugan, S. Bavuso and M. Boyd, "Dynamic Fault Tree Models for Fault Tolerant Computer Systems," *IEEE Transactions on Reliability*, Vol. 40, No. 3, pp. 363-377.

Glossary

Basic event. The bottom or "leaf" events of a fault tree. The limit of resolution of the fault tree. Examples of basic events are component failures and human errors.
Common cause failure. Multiple component faults that occur at the same time or that occur in a relatively small time window and that are due to a common cause.
Failure. An unacceptable deviation from the design tolerance or in the anticipated delivered service, an incorrect output, the incapacity to perform the desired function.
Fault. A defect, imperfection, mistake or flaw of varying severity that occurs within some hardware or software component or system. "Fault" is a general term and can range from a minor defect to a failure.
Fault realization or error. The manifestation of a fault in a system or the information that is processed by the system or a manifestation in the internal system state.
Minimal cut set. A smallest combination of basic events whose occurrence results in the occurrence of the top event of a fault tree.
Minimal path set. A smallest combination of basic events whose nonoccurrence results in the nonoccurrence of the top event of the success tree.
Permanent fault. A fault with lasting effects. The failed component or system must be replaced.
Recoverable, or covered, fault. A fault from which the system can recover all or part of its functionality. The system can continue to operate, although possibly in a degraded mode.
"State of component" fault. A fault of a component due to either the failure of the component or the failure of a command to the component.
"State of system" fault. A fault with a system-level effect and which is not necessarily localized at a given component.
Top event. The initial event of a fault tree or success tree. Also called the undesired event in the case of a fault tree.
Transient fault. A fault of limited duration that causes no permanent hardware damage. Transient faults can be caused by excessive heat, power disruptions, timing issues or environmental influences, for example. It is often possible to recover from a transient fault without discarding the affected component or system.
Undesired event. The top event of the fault tree.
Unrecoverable, or uncovered, fault. A fault from which the component or system cannot automatically recover. A unrecoverable fault leads to immediate system failure. Both transient and permanent faults can be unrecoverable.

Acronyms

AFTH	Fault Tree Handbook with Aerospace Applications
APU	Auxiliary Power Unit
BDD	Binary Decision Diagram
BE	Basic Event
CA	Containment Assurance
CCF	Common Cause Failure
DFM	Double Failure Matrix
DFT	Dynamic Fault Tree
ELV	Expendable Launch Vehicle
ESD	Event Sequence Diagram
ET	Event Tree
FDEP	Functional Dependency Gate
FHA	Fault Hazards Analysis
FMEA	Failure Modes and Effects Analysis
FMECA	Failure Modes, Effects and Criticality Analysis
FT	Fault Tree
FTA	Fault Tree Analysis
FV	Fussell-Vesely (Importance)
GUI	Graphical User Interface
HECS	Hypothetical Example Computer System
HRA	Human Reliability Analysis
HW	Hardware
IEEE	Institute for Electrical and Electronics Engineers
ISRB	Integrated Solid Rocket Booster
LOC	Loss of Crew
LOCV	Loss of Crew and Vehicle
LOM	Loss of Mission
LOV	Loss of Vehicle
MCS	Minimal Cut Set
mincut	Minimal Cut Set
minpath	Minimal Path Set
MIU	Memory Interface Unit
ML	Maximum Likelihood
MLD	Master Logic Diagram
MMOD	Micrometeoroid or Orbital Debris
MMSE	Minimum Mean Square Error
MS	Mission Success
MSE	Mean Square Error
MVUE	Minimum Variance Unbiased Estimator
NASA	National Aeronautics and Space Administration
NRC	Nuclear Regulatory Commission
PHA	Preliminary Hazards Analysis
PRA	Probabilistic Risk Assessment
RAW	Risk Achievement Worth (Importance)

RBD Reliability Block Diagram
RLV Reusable Launch Vehicle
RRW Risk Reduction Worth (Importance)
SAIC Science Applications International Corporation
SRB Solid Rocket Booster
SSME Space Shuttle Main Engine
ST Success Tree
SW Software
TVC Thrust Vector Control
VMS Vehicle Management System