

Mixing Reliability Prediction Models Maximizes Accuracy

Overcome Component Limitations, Better Reflect Past Experiences, and Achieve Superior Predictions

Although many models are available for performing reliability prediction analyses, each of these models was originally created with a particular application in mind. This document describes the most widely used reliability prediction models in terms of their intended applications, noting both their advantages and disadvantages. It then explains how mixing models in your reliability analyses yields more accurate predictions.

Widely Used Reliability Prediction Models

In any system, you have a mixture of electronic and mechanical parts. The selection of a reliability prediction model is driven by the critical parts in the system to be modeled and your system requirements. The following table lists the most widely used reliability prediction models and their intended applications, originating country, advantages, and disadvantages.

Reliability Prediction Model	Application & Originating Country	Advantages	Disadvantages
MIL-HDBK-217 The Military Handbook for the Reliability Prediction of Electronic Equipment	Military and Commercial, United States	<p>Provides for both Parts Stress and Parts Count analysis of electronic parts. Can easily move from preliminary design stage to complete design stage by progressing from Parts Count to Parts Stress.</p> <p>Includes models for a broad range of part types.</p> <p>Provides many choices for environment types.</p> <p>Well-known and widely accepted.</p>	<p>Is based on pessimistic failure rate assumptions.</p> <p>Does not consider other factors that can contribute to failure rate such as burn-in data, lab testing data, field test data, designer experience, wear-out, etc.</p> <p>NOTE: <i>The Relex Reliability Prediction module overcomes these limitations by allowing you to use Telcordia calculation methods and PRISM process grades with MIL-HDBK-217.</i></p>
Telcordia (Bellcore) Reliability Prediction Procedure for Electronic Equipment (Technical Reference # TR-332 or Telcordia Technologies Special Report SR-332)	Commercial, United States	<p>Offers analysis ranging from Parts Count to full Parts Stress through the use of Calculation Methods.</p> <p>Considers burn-in data, lab testing data, and field test data.</p> <p>Well-known and accepted.</p>	<p>Considers only electronic parts.</p> <p>Supports only a limited number of Ground Environments.</p> <p>Fewer part models compared to MIL-HDBK-217.</p> <p>Does not account for other factors such as designer experience, wear-out, etc.</p> <p>NOTE: <i>The Relex Reliability Prediction module overcomes these limitations by allowing you to use PRISM process grades with Telcordia.</i></p>

<p>Mechanical The Handbook of Reliability Prediction Procedures for Mechanical Equipment (NSWC-98/LE1)</p>	<p>Military and Commercial, United States</p>	<p>Provides for analyzing a broad range of mechanical parts (seals, springs, solenoids, bearings, gears, etc.)</p>	<p>Limited to mechanical parts.</p>
<p>CNET 93 Recueil de Données de Fiabilité des Composants Electroniques RDF 93 (UTE C 80-819)</p>	<p>Telecommunications, France</p>	<p>Fairly broad range of part types modeled.</p> <p>Provides unique handling of PCBs.</p>	<p>Considers only electronic parts.</p> <p>Only available in French.</p>
<p>RDF 2000 Recueil de Données de Fiabilité RDF 2000 (UTE C 80-810)</p>	<p>Telecommunications, France</p>	<p>Introduces a new approach to failure rate modeling.</p> <p>Considers cycling profiles and their applicable phases when determining failure rate.</p> <p>Provides unique handling of PCBs.</p>	<p>Considers only electronic parts.</p> <p>Cannot be mixed with other models because of the unique way in which failure rates are calculated.</p> <p>Very new, still gaining acceptance.</p>
<p>HRD5 The Handbook for Reliability Data for Electronic Components used in Telecommunication Systems</p>	<p>Telecommunications, United Kingdom</p>	<p>Similar to Telcordia.</p> <p>Fairly broad range of part types modeled.</p>	<p>Considers only electronic parts.</p> <p>Not widely used.</p>
<p>299B Chinese Military Standard GJB/z 299B</p>	<p>Military, China</p>	<p>Provides for both parts stress and parts count analysis.</p>	<p>Considers only electronic parts.</p> <p>Currently used primarily in China.</p> <p>Based on an older version of MIL-HDBK-217.</p> <p>Cannot model hybrids.</p>
<p>PRISM System Reliability Assessment Methodology developed by the Reliability Analysis Center (RAC)</p>	<p>Military and Commercial, United States</p>	<p>Incorporates NPRD/EPRD database of failure rates.</p> <p>Enables the use of process grading factors, predecessor data, and test or field data.</p>	<p>Small, limited set of part types modeled.</p> <p>Newer standard, still gaining acceptance.</p> <p>Considers only electronic parts.</p> <p>Cannot model hybrids.</p> <p>No reference standard available.</p>
<p>NPRD/EPRD Nonelectronics Parts Reliability (NPRD) and Electronic Parts Reliability (EPRD) databases by RAC</p>	<p>Military and Commercial, United States</p>	<p>Broad array of electronic and non-electronic parts.</p> <p>Based completely on field data.</p>	<p>Consists entirely of databases of failure rates, not mathematical models.</p>

Mixing Models to Overcome Component Limitations

Each reliability prediction model has its own set of advantages and disadvantages. By mixing the models used in your reliability analyses, you can greatly improve the accuracy of your predictions. For example, even very simple systems often have both electronic and mechanical components. To accurately predict the failure rates of both electronic and mechanical components, you would select a reliability model for electronic components, such as MIL-HDBK-217 or Telcordia, and also refer to *The Handbook of Reliability Prediction Procedures for Mechanical Equipment* from NSWC. By using both electronic and mechanical component models in your reliability analyses, you would obviously obtain more accurate predictions for the system and its components than by using either model alone.

The need to mix reliability prediction models for the electronic components in a system stems from limitations on the component types that these models support. For instance, suppose you select Telcordia as the basis for analyzing the reliability of your electronic components; then, during your analysis, you realize that Telcordia does not support some of the switches and relays used in your system. By adding MIL-HDBK-217 to your modeling mix, you would gain comprehensive coverage for switches, relays, and several other components not supported by Telcordia.

Similarly, if you selected PRISM as the basis for your analysis, coverage for switching devices, connectors, rotary devices, and inductors would be missing. To accurately assess system MTBF (Mean Time Between Failure) for systems with these components, you would have to add reliability models that covered these components to your modeling mix. Having multiple models available for your reliability analyses makes it much more likely that the failure rates predicted for the system and its component are accurate.

Mixing Models to Better Reflect Past Experiences

In addition to mixing reliability prediction models because of part type limitations, you may want to mix models because certain ones more accurately predict the failure rates your system components have experienced in the past. For example, perhaps the failure rates calculated by PRISM best reflect those for the integrated circuits in your system, and the failure rates calculated by Telcordia best reflect those for the resistors in your system. In such cases, you would want to be able to choose the model that calculates the failure rates closest to those experienced in the past for each type of system component. The ability to choose completely different models for various components in the same system empowers you to generate the most accurate predictions possible.

Mixing Techniques for Superior Predictions

NOTE: *The following paragraphs describe features that are applicable only to specific reliability prediction models. However, none of these limitations apply to the Relex Reliability Prediction module. Providing that you have licensed a model described in this document, the Relex Reliability Prediction module supports the use of that model's features with all other licensed models.*

Although PRISM has models for calculating the failure rates of only a limited number of components, it provides many techniques for enhancing reliability predictions. For example, you can use PRISM process grades, which explicitly account for factors contributing to system reliability by grading the process for each system failure cause. If you think the reliability of a component is affected by process-related variability during the design and manufacturing process, you can use process grades to adjust the failure rates calculated for those components.

PRISM also provides summary data from RAC's Nonelectronics Parts Reliability (NPRD) and Electronic Parts Reliability (EPRD) databases for estimating failure rates of components that do not have models. If some of your components are operating within a specific set of environmental conditions and quality levels, you can retrieve the actual life-based failure rate values for components in very similar operating conditions from the NPRD and EPRD databases and then use these values in conjunction with reliability prediction models.

PRISM also allows you to include empirical data on a predecessor system and test data or field data to update the predicted reliability values. Similarly, Telcordia offers Calculation Methods to take advantage of burn-in-data, lab testing data, or field test data that has been collected. If you have such data for certain components, you will want to take advantage of it in the modeling of these components.

In most cases, you would need to use PRISM to factor in process grades, empirical data on a predecessor system, and test data or field data to update predicted reliability values. Likewise, you would need to use Telcordia to have its Calculation Methods factor in burn-in, lab testing, and field test data. However, if you use the Relex Reliability Prediction module to perform your reliability analyses, such limitations do not exist. The Relex Reliability Prediction module extends the advantages and features unique to individual models to all models. Therefore, you can apply the process grade factors defined in PRISM to any licensed model to adjust the failure rates according to design and manufacturing factors. Or, the Calculation Methods defined in Telcordia for adjusting failure rates based on burn-in, lab testing, and field test data can be applied to any other licensed models.

In conclusion, having many reliability prediction models available for your use will help to accurately assess your system MTBF. You can select the model best suited to your specific system parameters and your individual needs.

Relex Reliability Prediction supports all of the models mentioned in this brief. If you would like additional information about how the Relex Reliability Prediction module provides for mixing models and techniques for superior results, please email info@relexsoftware.com.

Why FRACAS Means Superior Quality and Reliability

Closing the Loop to Improve Your Products and Processes

In reliability engineering, **FRACAS** is an acronym for **F**ailure **R**eporting, **A**nalysis, and **C**orrective **A**ction **S**ystem. FRACAS is the term used to designate a process by which companies track product defects and effectively respond and make corrections to fix problems. Product defects may be tracked during product design, testing, manufacturing, and field deployment. A comprehensive FRACAS allows you to efficiently track issues and ensure that reliability problems are addressed in a timely and successful manner. The FRACAS process is used throughout all types of industries. Successful FRACAS programs provide for:

- Easy and timely collection of accurate failure and maintenance data from the lab, field, and supply chain in standardized and compatible data structures.
- Total elimination of complicated and redundant paper-based data records throughout the failure collection process, thereby avoiding dual reporting.
- Effective data analysis to determine root cause mechanisms and real-time failure trends for fast and accurate decision-making.
- Accurate historical reliability performance measures, such as mean time between failures (MTBF), mean time to failure (MTTF), mean time to repair (MTTR), and availability for use in determining appropriate corrective actions.
- Customizable reports that facilitate and support corrective action decisions by both management and engineering personnel regarding the improvement of designs, manufacturing processes, and field support systems.
- Optimized workflow management for timely dissemination, information accessibility, and rapid feedback and approval cycles to fully support Collaborative Engineering, Six Sigma, and ISO initiatives.
- Continued monitoring and testing to ensure that implemented corrective actions either prevent failure recurrence or simplify or reduce maintenance tasks.
- Support of legacy systems and contribution to a common database for reliability, maintainability, and system component and part information.
- Automatic conveyance of all failure data and subsequent analysis results to product and system designers to drive design innovation.

Closed-Loop System

To provide such features to the various workgroup and enterprise levels that need them, a FRACAS must be an aggressive closed-loop system that is configurable, flexible, and scaleable! This means that all reported failures and faults must be entered in the FRACAS in an appropriate and controlled manner so that they can then be analyzed and corrective actions identified, implemented, and verified. The knowledge gained from this process must then be fed back into the design, manufacturing, and test process so that quality and reliability are improved. A simplified version of the closed-loop feedback path for a FRACAS follows.



A FRACAS formally captures predetermined types of data about a failure in an Incident Failure Report (IFR). Completed IFRs are submitted to analysts so that they can identify the corrective actions that are to be implemented to prevent these failures from recurring. Whenever corrective actions are implemented and verified or are otherwise determined to be unnecessary, the

IFR is closed. To reduce the possibility of an unmanageable backlog of open IFRs, management periodically reviews all unresolved IFRs to ensure their assignment and eventual closure.

A FRACAS can be used throughout the life cycle of any hardware or software product or system process to track failures, incidents, issues, and even enhancements or suggestions. By implementing a FRACAS during the initial design phase, significant cost savings can be realized from early problem correction, when even major design changes can still be considered to eliminate or reduce susceptibility to known failure causes. Additional benefits of implementing corrective actions while performing in-house tests and inspections are the many opportunities that exist for determining if these corrective actions adequately solve the reported issues.

If a FRACAS is not in place before product inspection or testing begins, problems often go totally unrecorded or insufficient data is captured. If a FRACAS is not in place before the product or process is put into production, it is unlikely that failure and maintenance data will be collected in a structured and timely manner. Consequently, determining and implementing effective corrective actions that either prevent failures from recurring or simplify maintenance tasks will become very difficult.

Failure Logging

All problems that occur during inspections, tests, and field use must be entered in the FRACAS using an established procedure for recording accurate failure information. Personnel who enter IFRs in the FRACAS should be properly trained to precisely capture the required data. To make entering data in IFRs easy, the entry forms for capturing failure information should be tailored to your hardware, software, or process. Once an IFR is created, the FRACAS should alert the responsible analyst to its existence and indicate the next required action.

Failure Analysis

An analyst examines the information entered in an IFR to determine the root cause of the failure and identify contributing factors. Methods for analyzing the root cause range from simple investigations of circumstances surrounding a failure to sophisticated laboratory analyses of failed parts. Once the analyst has established the root cause and contributing factors, he or she must develop logically derived corrective actions. As the number of IFRs in the FRACAS grows, the analyst can call upon the historic data for related or similar failures for help in resolving what the appropriate corrective actions for a failure should be. Once corrective actions are noted, the FRACAS should alert the technician who must perform them.

Corrective Action and Verification

When a technician is implementing corrective actions, he or she may be required to submit time or utilization logs containing operational hours and other time-related data needed to calculate MTBF. The technician may also be required to submit field service logs indicating maintenance times, actions, and part replacements. Visual monitoring or testing must then be performed to indicate that the corrective actions taken have either eliminated the failure or reduced its occurrence. Although verification is sometimes performed by the same or another technician, close out of the IFR is generally performed only by a manager.

Workflow Management

To be effective in meeting internal and external commitments, a FRACAS must provide for effectively managing the resources and strategies necessary to address open IFRs. Workflow management features within your FRACAS should facilitate the failure analysis process. Managers should be able to assign priorities to failures based on urgency, budgets, and the availability of personnel. To ensure that all IFRs are closed in a timely manner, managers should be able to track IFRs by priority levels, workflow resolution activities, resource assignments, and many other criteria. Management should also strive to improve quality and reliability by participating in the development, implementation, and verification of corrective actions and periodically reviewing failure trends.

Closing the Loop

A FRACAS builds upon and leverages all of the data entered in its centralized database to ensure early and sustained achievement of improved reliability and maintainability for your product or process. The structured procedure for entering, analyzing, and resolving IFRs produces valuable data about the reliability of your product or process and the efficiency of your organization to address issues. Analyzing and reporting on the IFRs in your FRACAS is vital to the efficiency and profitability of your organization. In addition to complete failure summary reports listing events and problems for specified time periods, your FRACAS should be able to generate root cause analyses (RCAs), problem analysis reports (PARs), material disposition reports (MDRs), product performance reports (which provide MTBF, MTTF, MTTR, availability, etc.), and failure trend charts. Feeding this critical information back into your design, manufacturing, and testing process provides the closed loop that promotes continuously improving quality and reliability throughout the life cycle of your product or process.

If you would like additional information about how the Relx FRACAS Management System can close the loop to dramatically improve the quality and reliability of your products and processes, please email info@relexsoftware.com.

Understanding Importance Measures in Fault Tree Analysis

Calculating Birnbaum, Criticality, and Fussell-Vesely Importance Measures

Reliability importance measures attempt to identify the fault tree event whose improvement will yield the greatest improvement in system performance. The three most popularly used importance measures are:

- Birnbaum
- Criticality
- Fussell-Vesely

This technical brief explains how to calculate these three importance measures and describes the underlying logic that led to their development. It also demonstrates how to use each importance measure by rank ordering the basic events by the values of their importance measures and then considering improving first that basic event with the highest importance measure value.

Birnbaum Importance Measure

The Birnbaum importance measure is defined as:

$$I_B(A) = P\{X|A\} - P\{X|\sim A\}$$

Where:

A indicates that the event whose importance is being measured occurred.

$\sim A$ indicates that this event did not occur.

X indicates the top event.

The Birnbaum importance measure for the event A is the difference in the probability of the top event given that the event A did occur minus the probability of the top event given that the event A did not occur. This is one measure of the increase in the probability of the top event due to the event A.

Consider a top event X, which is the result of event A and event B being connected by an OR gate. The fault tree would define the top event X to be $X = \{A \text{ or } B\}$. Assume that the probability of event A is 0.1 and that of event B is 0.2.

Let $P\{X|A\}$ denote the probability of the top event X given that the basic event A occurred. Clearly, if A occurs, $\{A \text{ or } B\}$ occurs, so that X occurs. Therefore:

$$P\{X|A\} = 1.0$$

Also, let $P\{X|\sim A\}$ denote the probability of the top event given that the basic event A does not occur. Here, given that A does not occur, X only occurs if the event B occurs. Therefore:

$$P\{X|\sim A\} = P\{B\} \text{ where } P\{B\} = 0.2$$

Thus, the Birnbaum importance measure equals:

$$I_B(A) = (P\{X|A\} - P\{X|\sim A\}) = (1.0 - P\{B\}) = (1.0 - 0.2) = 0.8$$

Criticality Importance Measure

Although the Birnbaum importance measure, $I_B(A)$, is useful, it does not directly consider how likely the event A is to occur. For instance, in the previous example, $I_B(A) = (1.0 - P\{B\})$ does not even involve the probability of the event A. This could lead to assigning high importance values to events that are very unlikely to occur and may be very difficult to improve. Remember, an event with a low probability of occurring in a fault tree is an event that has already been improved, so further improvement may be difficult to obtain. Therefore, in an attempt to focus only on those events that truly are important (which not only lead to the top event but also are more likely to occur and may reasonably be improved), a modified Birnbaum importance measure known as a Criticality importance measure is used.

The Criticality importance measure is defined as:

$$I_C(A) = I_B(A) * P\{A\}/P\{X\} \\ = (P\{X|A\} - P\{X|\sim A\}) * P\{A\}/P\{X\} \text{ where X is the top event.}$$

The Criticality importance measure modifies the Birnbaum importance measure by:

- Adjusting for the relative probability of the basic event A to reflect how likely the event is to occur and how feasible it is to improve the event (which makes it easier to focus on the truly important basic events).
- Conditioning on the occurrence of the top event X to restrict the measure to evaluating the effect of the basic event A, not the probability of the top event X (which makes it possible to compare basic events between fault trees).

Now, the Criticality importance measure, $I_C(A)$, for the earlier OR gate example, where $P\{A\} = 0.1$ and $P\{B\} = 0.2$, is to be calculated. The probability of the top event, the event $X = \{A \text{ or } B\}$, is first calculated:

$P\{X\}$ is the probability of the top event occurring.

$P\{A\}$ is the probability of event A occurring.

$P\{\sim A\}$ is the probability of event A not occurring.

$P\{A \text{ and } B\}$ is the probability of both events A and B occurring.

$P\{A \text{ or } B\}$ is the probability of either event A or event B or both events occurring.

If events A and B are independent, then $P\{A \text{ and } B\} = (P\{A\} * P\{B\})$. Therefore:

$$\begin{aligned} P\{X\} &= P\{A \text{ or } B\} \\ &= P\{A\} + P\{B\} - (P\{A\} * P\{B\}) \\ &= 0.1 + 0.2 - (0.1 * 0.2) = 0.28 \end{aligned}$$

Based on earlier calculations:

$$I_B(A) = 0.8, P\{A\} = 0.1, \text{ and } P\{X\} = 0.28$$

Therefore, the Criticality importance measure is given by:

$$\begin{aligned} I_C(A) &= (I_B(A) * P\{A\}) / P\{X\} \\ &= (0.8 * 0.1) / (0.28) = 0.2857143 \end{aligned}$$

Similar calculations for event B yields:

$$I_B(B) = 0.9$$

And:

$$\begin{aligned} I_C(B) &= (I_B(B) * P\{B\}) / P\{X\} = \\ &= (0.9 * 0.2) / (0.28) = 0.6428571 \end{aligned}$$

Now, consider the Criticality importance measure for the AND gate, where:

$$\begin{aligned} P\{A\} &= 0.1, P\{B\} = 0.2, \text{ and } P\{X\} = P\{A \text{ and } B\} \\ &= (P\{A\}) * (P\{B\}) = \\ &= (0.1) * (0.2) = 0.02, \text{ by independence of the basic events A and B.} \end{aligned}$$

Here:

$$\begin{aligned} P\{X|A\} &= P\{A \text{ and } B|A\} \\ &= P\{B\} * P\{X|\sim A\} \\ &= P\{A \text{ and } B|\sim A\} = 0.0 \end{aligned}$$

And:

$$I_B(A) = P\{X|A\} - P\{X|\sim A\} = P\{B\} - 0.0 = P\{B\}$$

Thus:

$$I_C(A) = (I_B(A) * P\{A\}) / P\{X\} = (P\{B\} * P\{A\}) / P\{X\} = P\{X\} / P\{X\} = 1.0$$

Similarly,

$$I_B(B) = P\{X|B\} - P\{X|\sim B\} = (P\{A\} - 0.0) = P\{A\}$$

So that:

$$I_C(B) = (I_B(B) * P\{B\}) / P\{X\} = (P\{A\} * P\{B\}) / P\{X\} = P\{X\} / P\{X\} = 1.0$$

Given independence of the basic events, all of the basic events under an AND gate will have the same Criticality importance measure. Thus, the Criticality importance measure is uninformative for AND gates.

Fussell-Vesely Importance Measure

The Fussell-Vesely importance measure is calculated quite differently than the Birnbaum or Criticality importance measures. It is constructed using minimal cut sets. A cut set is a set of basic events whose occurrence causes the top event to occur. A minimal cut set is a cut set that would not remain a cut set if any of its basic events were removed.

For example, the set of all the basic events is a cut set (or else the fault tree would be meaningless). If the fault tree consists of a single AND gate, then the cut set consisting of all the basic events is the only cut set and the minimal cut set. This is because all events leading into an AND gate must occur in order for the AND gate to be activated.

If the fault tree consists of a single OR gate, then the cut set consisting of all the basic events is not a minimal cut set unless there is only one basic event. This is because only one event leading into an OR gate needs to occur for the OR gate to be activated. In this case, any collection of basic events is a cut set. Therefore, given an OR gate, only those cut sets containing a single basic event are minimal cut sets.

Minimal cut sets are important in fault trees because they may be used to calculate the probabilities of events, including the top event. For example, the probability of the top event is given by the probability of the union of all the minimal cut sets.

Another interesting probability associated with the basic event A is the probability of the union of all minimal cut sets containing the basic event A. This is because the probability of the union of all minimal cut sets containing the basic event A is the probability that the top event is caused by a cut set containing the event A. This is a measure of the association of the basic event A with the top event X. It does not directly measure the probability that the top event X was caused by the basic event A, but it does indicate the potential importance of the basic event A.

A useful fact is that the probability of the union (OR) of sets is equal to the sum of the probabilities of the sets when the sets are mutually exclusive. If the sets are "nearly" mutually exclusive and, in addition, the basic events are independent and their probabilities are small, then this equality is approximately satisfied. For example, suppose that two minimal cut sets, C1 and C2, are given by $C1 = \{A \text{ and } B \text{ and } C\}$ and $C2 = \{A \text{ and } D\}$.

Then, exactly:

$$\begin{aligned} P\{C1 \text{ or } C2\} &= P\{C1\} + P\{C2\} - P\{C1 \text{ and } C2\} \\ &= P\{C1\} + P\{C2\} - \{P\{A \text{ and } B \text{ and } C\} \text{ and } \{A \text{ and } D\}\} \\ &= P\{C1\} + P\{C2\} - P\{A \text{ and } B \text{ and } C \text{ and } D\} \\ &= P\{C1\} + P\{C2\} - P\{A\} * P\{B\} * P\{C\} * P\{D\}, \text{ which is approximately equal to } P\{C1\} + P\{C2\} \\ &\text{ when the probability of each of the basic events is small.} \end{aligned}$$

This idea is used in calculating the Fussell-Vesely importance measure. This measure considers the ratio of the probability of the union of all minimal cut sets containing the basic event A, divided by the probability of the union of all minimal cut sets. In practice, the numerator is replaced by the approximating sum of the probabilities of all minimal cut sets containing the basic event A, and the denominator uses the exact calculation, which is simply the probability of the top event X.

With the Fussell-Vesely importance measure, the fact that there is only one cut set for an AND gate leads to the uninformative result that all of the basic events leading to an AND gate will have the same value for the Fussell-Vesely importance measure. Now, consider the previous example of the fault tree with an OR gate. Two minimal cut sets exist: $C1 = \{A\}$ and $C2 = \{B\}$. Recall that $P\{A\} = 0.1$, $P\{B\} = 0.2$, and that $P\{X\} = P\{A \text{ or } B\} = 0.28$. Note that $C1$ is the only minimal cut set containing the basic event A, and $C2$ is the only minimal cut set containing the basic event B. Also, $P\{C1\} = P\{A\} = 0.1$, and $P\{C2\} = P\{B\} = 0.2$. Therefore, the Fussell-Vesely importance measures for the basic events A and B are given by:

$$I_{FV}(A) = P\{C1\} / P\{X\} = 0.1 / 0.28 = 0.3571429$$

$$I_{FV}(B) = P\{C2\} / P\{X\} = 0.2 / 0.28 = 0.7142857$$

Importance Measure Usage

If all three importance measures yield the same rank ordering of basic events, then the strategy for using the importance measures is straightforward. However, when the three importance measures yield different rank orderings of basic events, the following guidelines suggest how to select an appropriate solution:

- Keep in mind that the goal is to assist in selecting the next basic event to consider for improvement. It cannot be concluded definitively that a particular basic event must receive the next improvement effort.
- Ensure the correct **primary time point** is chosen. The rank ordering of the importance measures may be different at different time points.
- Consider averaging the three-way ranking of the three importance measure rank orderings for each basic event. This may indicate a consensus of the three measures.
- Study the sequential ranking, first by ranking by the Fussell-Vesely or Criticality importance measure and then break ties within the ranking by the Birnbaum importance measure.
- When in doubt or when calculation performance is an issue, the Criticality importance measure is probably a reasonable single measure to use. It considers the probability of the basic event (an improvement over the Birnbaum importance measure). However, if this is a problem with uninformative results caused by AND gates (as can happen with the Fussell-Vesely importance measure), then use the Birnbaum importance measure.

Calculating MTTF When You Have Zero Failures



at Docu

Reliability Growth

Observing and Predicting Trends in Reliability

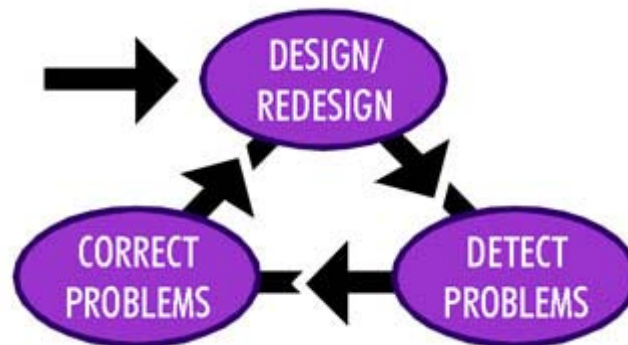
Today's complex products require that companies focus on establishing detailed plans and procedures for developing and manufacturing products that meet specified reliability, maintainability, and other performance requirements. **Reliability growth** refers to a well-defined process for identifying and correcting reliability problems early in the design process so that the reliability of a product increases or "grows" as the product goes through successive development stages. Reliability growth programs should be established for all new products and for all existing products undergoing major redesigns so that the improvement due to changes in design and manufacturing processes can be easily tracked.

During the early design stage, a reliability goal is set for a product. Because failure data from prototype testing is not yet available, the initial reliability goal is often based on either the failure data for similar products or the failure data for the subcomponents of the product. During the development stage, product prototypes generally undergo extensive testing so that deficiencies in respect to design, engineering, and manufacturing can be identified and corrected.

A typical reliability improvement test consists of operating product prototypes for several weeks in the same types of environments in which customers will eventually operate the product. A team of project engineers and technicians analyze every failure that occurs, determining root causes for failures and developing design and manufacturing improvements that are to either eliminate or reduce the recurrences of these failures. As the testing continues, the improvements developed by the team are incorporated into the prototype so that product reliability continues to improve throughout the testing period.

Achieving Reliability Growth

The three most important steps in the iterative process for achieving reliability growth are depicted in the feedback loop that follows:



Most of the problems encountered during testing are likely to be component failures and manufacturing deficiencies that could not be foreseen in the early design phase. Because various performance requirements can conflict, optimizing a design to meet one requirement can cause the design to fail to meet another requirement. Thus, iterations of designs are often needed to correct all of the component selection and manufacturing deficiencies that are found during prototype testing.

In addition to improving reliability, early implementation of a reliability growth program minimizes the impact on production scheduling and total product cost, especially since the costs associated with either redesigning a product late in the development cycle or retrofitting products already in the field are extremely high. Developing a product that meets reliability requirements also ensures that the product ultimately meets user needs and has an acceptable total life cycle cost. Setting interim reliability goals that are to be achieved during testing ensures that resources are allocated efficiently.

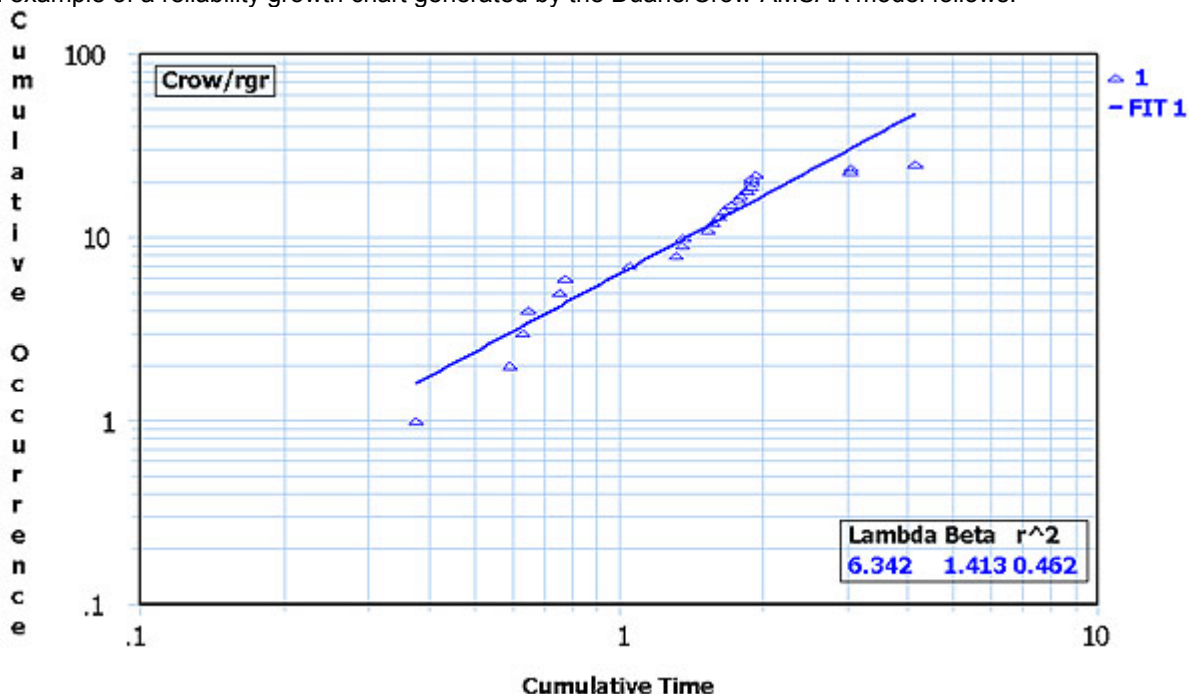
Reliability Growth Data

The basic principle of any reliability growth model is to apply the testing results and data points to determine if the reliability of the product is growing sufficiently to meet the reliability requirements for the product. The types of data used for predicting reliability growth are:

- **Reliability Data.** The reliability of the product is recorded at different points in time. The reliability is a ratio of the number of units still functioning and the number of units that entered the stage.
- **Success/Failure Data.** The item is tested and can either succeed or fail. Success data can consist of a code indicating the outcome, such as S for Success or F for Failure; or, it can consist of a code indicating the failure mode for each failure. Success/failure data might also indicate the number of failures that occurred when several units are tested together.
- **Failure Time Data.** The time to failure for an item is tracked using either cumulative or non-cumulative operating times. Failure time data is the most commonly used type of data in reliability growth. This data can be for one failure or system, or it can be for multiple failures or multiple systems, which is commonly called interval or grouped data.

The failure data collected during prototype testing is used to determine whether the reliability goal for the product is likely to be met or exceeded by the time the product is scheduled to be put into full-scale production. Although many methods exist for modeling the reliability growth process, the Duane/Crow-AMSAA model is considered the best practice. In 1964, J. T. Duane, an engineer at the Aerospace Electronics Department of General Electric Company, published a paper demonstrating how a learning curve approach could be used to monitor the continuing reliability improvements in the early stages of developing complex electromechanical and mechanical systems. Duane explained how this was because the lessons learned from failures were used to refine designs.

According to Duane, a graph of the cumulative MTBF versus the cumulative operating time plotted on log-log paper fell close to a straight line. Graphing this learning curve provides a means of measuring and predicting reliability during a period of product change. Dr. Larry Crow later added powerful statistical capabilities to Duane's postulate for learning curve modeling to create the model so widely respected and implemented today. An example of a reliability growth chart generated by the Duane/Crow-AMSAA model follows:



Reliability growth is generally quantified by graphing any of the following three measures over time:

- The increase in the mission success probability (reliability).
- The increase in MTBF.
- The decrease in failure rate as a function of time.

Reliability growth charts depict trends that are used to forecast failures as a function of additional test time or calendar time, thereby making planning for redesign and test resources easier. In addition to allowing you to determine whether reliability requirements will be achieved, reliability growth charts can help you to determine

the time needed to meet these requirements and the associated costs. Extrapolating a growth curve beyond the currently available data shows what reliability a program can be expected to achieve providing that the conditions of the test and the engineering effort to improve reliability are maintained at their present levels. If the reliability growth graph indicates that the reliability goal is not going to be met or exceeded, then the product design must be improved. This might require the use of more reliable components or redundancy. Additional resources might also have to be devoted to designing and manufacturing a more reliable product to meet the required delivery date.

Other Uses for Reliability Growth Charts

In addition to analyzing the results from testing newly developed or redesigned products, the Duane/Crow-AMSAA model can be used to:

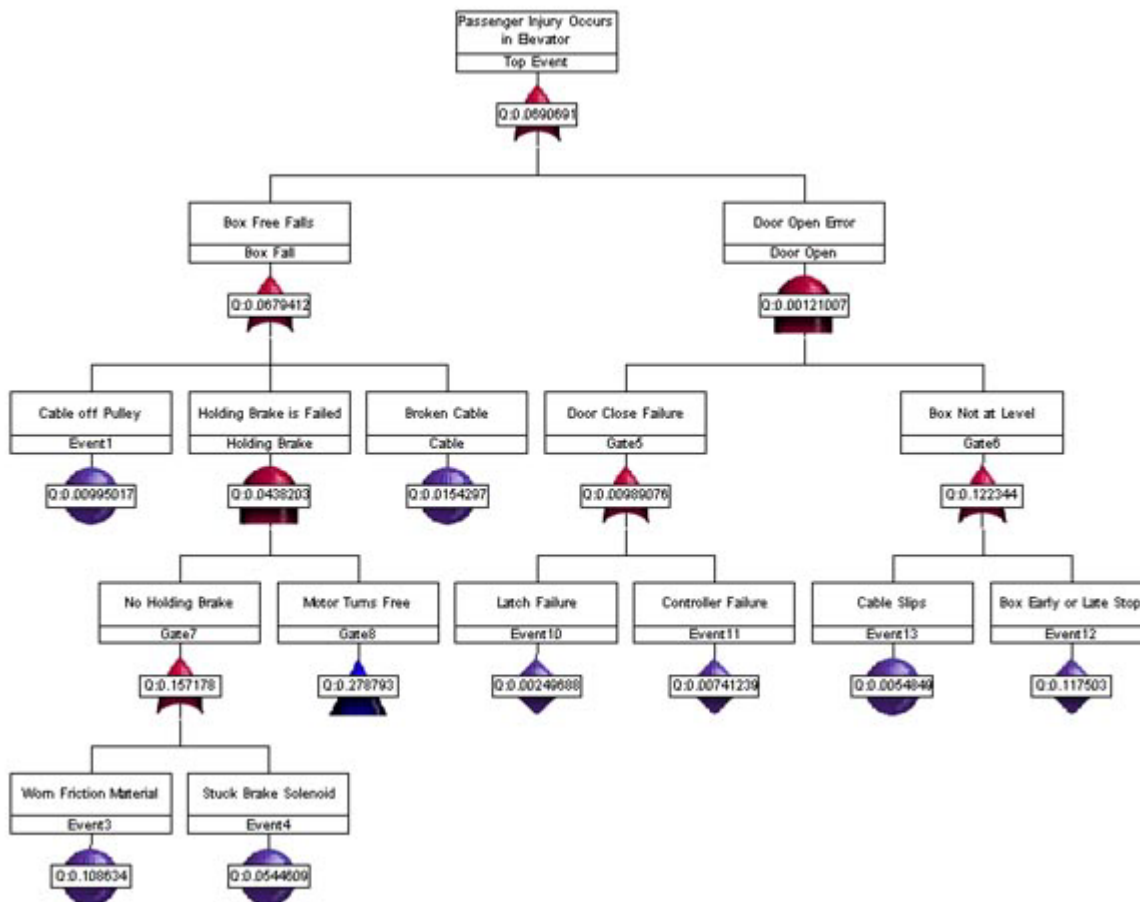
- Predict future failures.
- Track fleets of repairable systems.
- Provide accurate trending of significant events for management.
- Forecast safety incidents that can be controlled.

Fault Tree Gates Logically Link Basic Events to Top Events

Analyze Numerous Hardware Configurations Using Multiple Gate/Event Options

Fault Tree Analysis (FTA) is well recognized worldwide as an important tool for evaluating safety and reliability in system design, development, and operation. Based on a simple set of rules and logic symbols from probability theory and Boolean algebra, FTA uses a top-down approach to generate a logic model that provides for both qualitative and quantitative evaluation of system reliability. The undesirable event at the system level is referred to as the **top event**. It generally represents a system failure mode or hazard for which predicted reliability data is required. The lower level events in each branch of a fault tree are referred to as **basic events**. They represent hardware, software, and human failures for which the probability of failure is given based on historical data. Basic events are linked via logic symbols (**gates**) to one or more undesirable top events.

Today, computerized FTA is used to analyze very complex systems as well as very complex relationships between hardware, software, and humans. Small fault trees have fewer than 100 events, medium fault trees have from 100 to 1,000 events, and large fault trees have more than 1,000 events! Using good FTA software, you can cut, copy, paste, rearrange, and delete events and gates in various fault tree branches to quickly and easily compare different hardware configurations. An example of a very simple computer-generated fault tree follows.



* Generated in Relx Fault Tree. Click the image to view a full-size version.

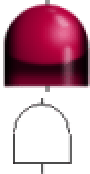

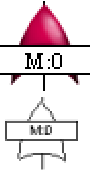

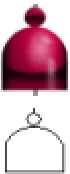
In this fault tree, "Passenger Injury Occurs in Elevator" is defined as the top event. The reasons why passenger injury in an elevator could occur have been determined to be either that the box free falls or that the door is open at an inappropriate time. After determining all possible causes for each event identified, the events and gates for connecting them to higher-level events are added to the fault tree. Any faults that can be further developed to determine causes are then added as lower-level events and connected by the appropriate gates.








The lowest-level basic events that terminate fault tree paths are often called **terminal events** or **primary events**. They are either component-level events that cannot be further resolved or external events. For example, in the first level of possible


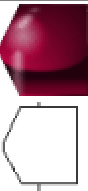

events for the free fall of the box, "Cable off Pulley" and "Broken Cable" are terminal events. Because these events are primary faults, they are not developed any further in the fault tree.

The tables below describe the fault tree gates and events that are implemented in Relex and many other computerized FTA programs. **Note:** Relex Fault Tree is the only commercial software product for reliability analysis that supports dynamic gates (Functional Dependency, Sequence-Enforcing, and SPARE gates). Although a few other competing products support two-input Priority AND gates, none of them perform the dynamic analysis (Markov) required.

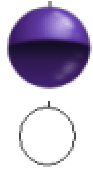



Fault Tree Gates

Bitmap/Line Art	Gate Name	Gate Description
	AND Gate	<p>The AND gate is used to indicate that the output occurs if and only if all the input events occur. The output of an AND gate can be the top event or any intermediate event. The input events can be basic events, intermediate events (outputs of other gates), or a combination of both. There should be at least two input events to an AND gate.</p> <p>Summary of Logic: All events must be TRUE for the output to be TRUE.</p>
	OR Gate	<p>The OR gate is used to indicate that the output occurs if and only if at least one of the input events occur. The output of an OR gate can be the top event or any intermediate event. The input events can be basic events, intermediate events, or a combination of both. There should be at least two inputs to an OR gate.</p> <p>Summary of Logic: If at least 1 event is TRUE, the output is TRUE.</p>
	Voting Gate (m/n)	<p>The Voting gate (m/n) is used to indicate that the output occurs if and only if m out of the n input events occurs. The m input events need not occur simultaneously. The output occurs when at least m input events occur. When m = 1, the Voting gate behaves like an OR gate. The output of a Voting gate can be a top event or an intermediate event. The input events can be basic events, intermediate events, or combinations of both.</p> <p>Summary of Logic: If m = 2 and n = 3, 2 input events must be TRUE for the output to be TRUE.</p>
	Exclusive OR Gate (XOR Gate)	<p>The Exclusive OR (XOR) gate is used to indicate that the output occurs if and only if one of the two input events occurs and the other input event does not occur. The output of an XOR gate can be the top event or an intermediate event. The input events can be basic events, intermediate events, or combinations of both. An XOR gate can have only two inputs.</p> <p>Summary of Logic: If 1 and only 1 input event is TRUE, the output is TRUE.</p>
	NAND Gate	<p>The NAND gate functions like a combination of an AND gate and a Not gate. The NAND gate is used to indicate that the output occurs when at least one of the input events is absent. The output of a NAND gate can be the top event or an intermediate event. The input events can be basic events, intermediate events, or combinations of both. The presence of a NAND gate may give rise to non-coherent trees, where the non-occurrence of an event causes the top event to occur.</p> <p>Summary of Logic: If there is at least 1 FALSE event, the output is TRUE.</p>

	NOR Gate	<p>The NOR gate functions like a combination of an OR gate and a Not gate. The NOR gate is used to indicate that the output occurs when all the input events are absent. The output of a NOR gate can be the top event or an intermediate event. The input events can be basic events, intermediate events, or combinations of both. The presence of a NOR gate may give rise to non-coherent trees, where the lack of an event causes the top event to occur.</p> <p>Summary of Logic: If there is at least 1 TRUE input event, the output is FALSE.</p>
	Not Gate	<p>The Not gate is used to indicate that the output occurs when the input event does not occur. The presence of a Not gate may give rise to non-coherent trees, where the non-occurrence of an event causes the top event to occur. There is only one input to a Not gate.</p> <p>Summary of Logic: The output is the opposite of the input gate or event.</p>
	Inhibit Gate	<p>The Inhibit gate is used to indicate that the output occurs when the input events (I1 and I2) occur and the input condition (C) is satisfied. The output of an Inhibit gate can be a top event or an intermediate event. The input events can be basic events, intermediate events, or combinations of both.</p> <p>Summary of Logic: If all input events and the input condition are TRUE, the output is TRUE.</p>
	Priority AND (PAND) Gate	<p>The Priority AND (PAND) gate is used to indicate that the output occurs if and only if all input events occur in a particular order. The order is the same as that in which the inputs events are connected to the PAND gate from left to right. The PAND gate is a dynamic gate, which means that the order of the occurrence of input events is important to determining the output. Relex PAND gates support multiple inputs. This is a generalization of an earlier version of the Relex PAND gate where only two inputs were allowed.</p> <p>The output of a PAND gate can be the top event or an intermediate event. The inputs can be basic events or outputs of any AND gate, OR gate, or dynamic gate (Priority AND gate, Functional Dependency gate, Sequence-Enforcing gate, or SPARE gate). (These gates should have the inputs from basic events or other AND gates and OR gates.) You may rearrange items that enter PAND gates to fail in temporal order from left to right to trigger the event. The PAND gate also supports a single input. When only a single input exists, then occurrence of that input will trigger the event.</p> <p>Summary of Logic: All input events must be TRUE for the output to be TRUE, and the events should occur from left to right in the temporal order.</p>
	Transfer Gate	<p>A Transfer gate is a symbol used to link logic in separate areas of a fault tree. There are two primary uses of Transfer gates. First, an entire fault tree may not fit on a single sheet of paper (or you may want to keep the individual trees small to view and organize them). Second, the same fault tree logic may be used in different places in a fault tree. Through the use of Transfer gates, you can define this logic once and use it in several places. To use a Transfer gate, you insert a Transfer In gate in a fault tree, that links to a Transfer Out gate, which represents the top gate of another fault tree.</p>
	Remarks Gate	<p>A Remarks gate is used for the entry of comments. A Remarks gate has no calculation data associated with it, and, therefore, has no effect on calculations. However, the tree branch may continue after a Remarks gate. There can only be one input to a Remarks gate.</p>
	Pass-Through Gate	<p>A Pass-Through gate is used for visually aligning the events and gates in a fault tree. A Pass-Through gate extends a vertical connector for visual alignment. A Pass-Through gate has no calculation data associated with it, and, therefore, has no effect on calculations. However, the tree branch may continue after a Pass-Through gate. There can be only one input to a Pass-Through gate.</p>

	<p>Functional Dependency Gate</p>	<p>The functional dependency (FDEP) gate is used to indicate that all dependent basic events are forced to occur whenever the trigger event occurs. The separate occurrence of any of the dependent basic events has no effect on the trigger event. The FDEP gate has one trigger event and can have one or more dependent events. All dependent events are either basic events or spare events. The trigger event can be a terminal event or outputs of any AND gate, OR gate, or dynamic gate (PAND gate, FDEP gate, Sequence- Enforcing gate, or SPARE gate).</p> <p>Dependent events are repeated events that are present in other parts of the fault tree. The FDEP gate is a dynamic gate, which means the temporal order of the occurrence of events is important to analyze this gate. Generally, the output of the FDEP gate is not that important; however, it is equivalent to the status of its trigger event.</p> <p>The FDEP gate can also be used to set the priorities for SPARE gates. For example, if multiple spares are connected to a FDEP gate, after the occurrence of the trigger event, all spares that are connected to the FDEP gate will fail. Upon failure of these spares, the next available good spare in those SPARE gates will replace the failed spares. If there exists a conflict in choosing the next available spare between multiple SPARE gates, the priority will be based on the order of the connection of these spares in the FDEP gate from left to right.</p> <p>Summary of Logic: When the trigger event is TRUE, then dependent events are forced to become TRUE. The trigger event must be TRUE for the output to be TRUE.</p>
	<p>Sequence-Enforcing Gate</p>	<p>The sequence-enforcing (SEQ) gate forces events to occur in a particular order. The input events are constrained to occur in the left-to-right order in which they appear under the gate. This means that the leftmost event must occur before the event on its immediate right, which must occur before the event on its immediate right is allowed to occur, etc. The SEQ gate is used to indicate that the output occurs if and only if all input events occurs, when the input events must occur in a particular order.</p> <p>The SEQ gate is a dynamic gate, which means the occurrence of the inputs follows a sequential order. In other words, an event connected to a SEQ gate will be initiated immediately after occurrence of its immediate left event. Therefore, if the leftmost input is a basic event, then the SEQ gate works like a cold SPARE gate. The SEQ gate can be contrasted with the PAND gate in that the PAND gate detects whether events occur in a particular order (but the events can occur in any order), whereas the SEQ gate allows the events to occur only in the specified order.</p> <p>The first input (leftmost input) to a SEQ gate can be a terminal event or outputs of any AND gate, OR gate, or dynamic gate (PAND gate, FDEP gate, SEQ gate, or SPARE gate). Only basic events are allowed for all other inputs. You may rearrange the events that enter SEQ gates; however, rearrangement is not allowed if the first input is an output from another gate.</p> <p>Summary of Logic: The output is TRUE if and only if all input events are TRUE; but the input events must occur in a particular order.</p>
	<p>SPARE Gate</p>	<p>The SPARE gate is used to model the behavior of spares in the system. The SPARE gate is used to indicate that the output occurs if and only if all input spare events occur. All inputs of a SPARE gate are spare events. A SPARE gate can have multiple inputs. The first event (leftmost event) is known as the primary input, and all other inputs are known as alternative inputs. The primary event is the one that is initially powered on, and the alternative inputs specify that they are in standby mode. After a failure, the active/powered unit that is the first available spare from left to right will be chosen to be active. If all units are failed, then the spare will be considered as failed (output occurred).</p> <p>Depending on the dormancy factor of spares, spares can fail even in standby mode. The Relex SPARE gate is more flexible and can handle any kind of spares. If the dormancy factor of all spares connected to a SPARE gate is 0, then the spare acts like a cold spare. If the dormancy factor of all spares connected to a SPARE gate is 1, then the spare acts like a hot spare. If the dormancy factors of all spares connected to a SPARE gate are the same (and are between 0 and 1), then the spare acts like a warm spare. If the dormancy factors of its inputs are different, then it handles generalized situations. The SPARE gate is a dynamic gate, which means the temporal order of the occurrence of events is important to analyze this gate. You may rearrange spare events that enter SPARE gates.</p> <p>Summary of Logic: All inputs must be TRUE for the output to be TRUE.</p>

Fault Tree Events

Bitmap/Line Art	Event Name	Event Description
	Basic Event	<p>A Basic event is either a component-level event that is not further resolved or an external event. It is at the lowest level in a tree branch and terminates a fault tree path. Component-level events can include hardware or software failures, human errors, and system failures.</p>
	Spare Event	<p>A Spare event is used to specify spares in dynamic fault trees. A Spare event is similar to a basic event in functionality; however, a spare event allows only rates as inputs. The dormancy factor of the spare indicates the ratio of failure rate in the spare mode and the failure rate in the operational mode. Spare events can have a spare pool, which represents the number of identical instances of that event. For example, if a spare pool of an event is 2, there are 2 identical spare components of that spare event. Spare events are restricted to use as either spares to SPARE gates or as dependent events to FDEP gates.</p>
	House Event	<p>A House event can be turned on or off. When a House event is turned on (TRUE), that event is presumed to have occurred and the probability of that event is set to 1. When a House event is turned off (FALSE), it is presumed not to have occurred, and the probability is set to 0. House events are useful in making parts of a fault tree functional or non-functional. When a House event is turned off, the gate that the House event inputs to will be removed from the tree during calculation. By turning that same House event on, the gate that the House event inputs to will be calculated normally. House events are also referred to as trigger events and switching events.</p>
	Undeveloped Event	<p>An Undeveloped event is used if further resolution of that event does not improve the understanding of the problem, or if further resolution is not necessary for proper evaluation of the fault tree. It is similar to a Basic event, but is shown as a different symbol to signify that it could be developed further, even though you have not done so for the analysis. Undeveloped events may be broken down into associated gates and events.</p>

Selecting the Appropriate Reliability Model

Choosing Between RBDs, Fault Trees, Event Trees, and Markov Analyses

System failure depends on the combination and sequence of component failures. If the system failure can be expressed completely based on the combinations of component failures, then the failure model is known as a combinatorial model.

To model any system effectively, you must:

- Understand the overall system configuration and system behavior.
- Be able to define the failure and repair distributions of the components.
- Be able to represent the system structure visually.

The above system knowledge is essential to determining and performing the appropriate mathematical analysis.

The table below provides descriptions of several modeling methodologies and lists both the advantages and disadvantages associated with their traditional implementations. It then notes the more recent advances that either reduce or eliminate some of these disadvantages.

Note: Traditional RBDs, Fault Trees, and Event Trees cannot model the temporal order of events. This means that, in these methodologies, it does not matter whether component 1 fails first or component 2 fails first. If component 2 is used only after component 1 fails, then the order of events is important. Only Markov models can consider the order of events and the actual times of their occurrences. Thus, Markov models are the only correct models for highly dependable, complex systems.

	RBDs	Fault Trees	Event Trees	Markov Analyses
Description	Models system success logic using modular or block structure.	Models system level faults, external events, conditional events, and non-coherence. Primarily used for safety analysis in systems where the probability of critical failures can be very low.	Models consequences and are often used with Fault Trees to model complex behaviors.	Models complex scenarios such as repairs, priorities, standbys, and shared loads.
Advantages	<p>Easy to understand.</p> <p>Easy to evaluate using analytical methods.</p> <p>Easy to integrate various components using sub-models/blocks.</p> <p>Easy to model redundancy.</p> <p>Easy to specify component priorities.</p> <p>Easy to verify system success paths.</p>	<p>Events can be generic faults (meaning that they need not be failures).</p> <p>Can be modular.</p> <p>Easy to visualize the cut sets.</p> <p>Easy to understand system level failures.</p> <p>Are evolutionary, allowing details to be added.</p> <p>Can be used with Event Trees to model</p>	<p>Can model various failure modes and their associated consequences.</p> <p>Used for risk and safety analysis.</p> <p>Can be used to model capacity.</p>	<p>Provides flexibility.</p> <p>Models temporal orders of events, making the analysis of complex scenarios possible.</p> <p>Considers state dependent transition rates (failure and repair).</p> <p>Can specify capacities on states. (Capacities are also called rewards.)</p>

		complex behaviors.		
Disadvantages	<p>Models only one type of system failure mode/consequence, combining all failures as one.</p> <p>Can model only failure and repair events, thereby excluding human errors and external events from consideration. (Otherwise, you must use "fictitious" components.)</p> <p>Models only coherent structures.</p> <p>Computational time can be lengthy for complex models.</p>	<p>Models are static, providing no method for standby comparisons.</p> <p>Priorities cannot be set (unless dynamic gates are used).</p> <p>Modeling capacity is only possible using multiple Fault Trees.</p> <p>Computational time can be lengthy for non-modular Fault Trees.</p>	<p>Models are static, providing no method for standby comparisons.</p>	<p>Supports only exponential distributions.</p> <p>Models are more difficult to construct and harder to understand.</p> <p>Computational time can be lengthy.</p>
Methodology Advances	<p>Switches and junctions are extending RBDs to non-modular (network) applications.</p> <p>Flow and capacity can now be considered.</p> <p>Switches, delays, and priorities are allowing for modeling temporal and conditional events.</p> <p>Standby redundancies provide for modeling temporal events.</p>	<p>The addition of dynamic gates now allow Fault Trees to model temporal events, spares, priorities, sequences, and forced failures.</p>	<p>Limited acceptance of dynamic Event Trees. Therefore, these techniques are not available in commercial software products.</p>	<p>Semi-Markov and non-homogeneous Markov models can handle highly complex systems, including all types of failure distributions. However, users must have good modeling skills to use these methodologies.</p>

FRACAS: Providing Continual Improvement in Quality and Reliability

Creating an Effective FRACAS for Any Application

A FRACAS (Failure Reporting, Analysis, and Correction Action System) is an important process by which the quality and reliability of a product, service, process, or software application can be tracked, measured, and ultimately improved. Companies known for providing highly reliable products have identified a comprehensive closed-loop FRACAS as one of the most critical elements in their reliability programs. The following table, published by the Reliability Analysis Center¹ (RAC), is based on a 1995 survey of reliability tasks.

Most Important Reliability Tasks Based on Normalized Score

Rank	Task	Normalized Score
1	FRACAS	88.3
2	Design Reviews	83.8
3	Subcontractor/Vendor Control	72.1
4	Parts Control	71.2
5	FMECA	70.3
6	Reliability Qualification Test	68.5
7	Predictions	62.2
8	Test, Analyze and Fix (TAAF)	59.5
9	Thermal Analysis	58.6
10	ESS	54.1

As you can see, 88.3 percent of the survey respondents viewed a FRACAS as the most important reliability task. It received the highest rating because of its ability to feed root failure cause and corrective action information back into the design process to further improve design reliability. And, as RAC points out, early elimination of root failure causes greatly contributes to both product reliability growth and attaining customer satisfaction following product/service delivery.

FRACAS Applications

A FRACAS can be adopted to cover a wide range of applications, including but not limited to:

- A FRACAS for hardware, software, and processes.
- A FRACAS for any kind of product (such as electronic boxes, airplane development or manufacturing, airlines, cellular telephone service, software development, production lines, services, etc.).
- A FRACAS for the entire product life cycle (such as development, prototypes, testing, field testing, service, storage, maintenance, etc., or, for software development, integration, field service, or other any conceivable phase in the product life).
- A FRACAS answering various civilian and military specifications, such as FAA, US DOD, IEC, NATO, IEEE, SAE, or other national or organization requirements.

The *Failure Reporting, Analysis and Corrective Action System (FRACAS) Application Guidelines*² provides information about what a FRACAS is, how it can be effectively tailored, and how it can be applied beyond the traditional tracking of hardware failures. To perform a qualitative assessment, such as whether a customer is satisfied with a product or service, or a quantitative assessment, such as whether the reliability performance goals of a product or service have been achieved, a FRACAS must be

used consistently. According to this publication, defining what constitutes a failure is perhaps the most critical aspect of implementing an effective FRACAS. The following table provides RAC definitions for a failure and failure events.

Failure and Failure Events

Terms	Definitions
Failure	An event in which an item does not perform one or more of its required functions within the specified limits under specified conditions. A failure can either be catastrophic (total loss of function) or out-of-tolerance (degraded function beyond specified limits due to such occurrences as part failure, detuning, misalignment, and maladjustment, which are often classified as faults).
Failure Symptom	Any circumstance, event, or condition associated with the failure that indicates its existence or occurrence. Failure symptoms can include a temporary, intermittent indication of failure that cannot be duplicated.
Failure Effect	The consequence that a particular failure mode has upon the operation, function, or status of a product or service.
Failure Mode	The type of defect contributing to a failure, the consequence of the failure (i.e., how the failure manifests), or the manner in which the failure is observed.
Failure Mechanism	The process that results in the failure; the process of degradation or chain of events leading to and resulting in a particular failure mode.
Failure Cause	The circumstance that induces or activates a failure mechanism, e.g., defective soldering, design weakness, assembly techniques, software error, manufacturing process, clerical error, etc.

Failure Classifications

According to the *Failure Reporting, Analysis and Corrective Action System (FRACAS) Application Guidelines*, the classification of failure events helps to determine:

- The level of analysis that should be performed on each failure.
- The appropriate corrective action that should be taken (and when) to eliminate the failure or at least to minimize its recurrence.

The following table lists the generic categories that RAC uses to classify failures.

Classification	Definitions
Failure, Relevant	A product (or service) failure that has been verified and can be expected to occur in normal operational use. Relevancy indicates whether a specific failure should "count" or not in the calculation of reliability for a produce or service (see "Failure, Chargeable" below).
Failure, Non-Relevant	A product (or service) failure that has been verified as having been caused by a condition not defined for normal operational use.
Failure, Chargeable	A relevant primary failure of the product (or service) under test, and any secondary failures resulting from a single failure incident. This definition of failure is typically limited to formal, contractually required reliability tests (performed in-house or in the field).

Failure, Non-Chargeable	A non-relevant failure, or a relevant failure caused by a previously agreed to set of conditions that eliminates the assignment of failure responsibility to a specific functional group. This definition of failure is typically limited to formal, contractually required reliability tests (performed in-house or in the field).
Failure, Pattern	The occurrence of two or more failures of the same part (or function) in identical or equivalent applications, where the failures are caused by the same basic failure mechanism, and the failures occur at a rate inconsistent with the expected part (or function) failure rate.
Failure, Multiple	Simultaneous occurrence of two or more verified independent failures. When two or more failed parts are found during troubleshooting, and assignable causes cannot be verified as dependent, multiple failures are presumed to have occurred.

In developing a FRACAS, you may choose to either adopt the RAC terminology and classifications or to use your own.

Markov Analysis, Part 1

Markov Analysis Accurately Models Dynamic Behaviors

Combinatorial models such as reliability block diagrams (RBDs) and fault trees are used to predict the reliability of complex systems. However, they cannot accurately model such dynamic system behaviors as:

- Repairs.
- Common-cause and dependent failures.
- Shocks (shared loads and induced failures).
- Sequence/state-dependent failure rates (standby components).
- Variable configurations.
- Complex error handling and recovery mechanisms (common pool of repair technicians).
- Phased mission requirements.

Because of their flexibility, generalized **stochastic processes** are widely used to assess system reliability and related characteristics in mission critical systems.

Stochastic Processes

Stochastic processes have a number of states that describe the behavior of a set of random variables. The behavior of the stochastic process varies with respect to an index. In reliability engineering, this index is generally system time. This means that the stochastic process is used to describe the dynamics of a system with respect to time.

State space is the set of all possible states of a process, and **index space** is a set of all possible index values. At a particular time (index value), a system will be in one of its possible states. In each state, a set of events can occur. The occurrence distribution of each state depends on the history of the system (all previous events and state transition times).

In reliability engineering, the state space is generally discrete. For example, a system might have two states: good and failed. There are, however, applications in which state space can be continuous. Examples include the water level in a tank (where tank failure characteristics depend on the water level), the load on a shaft, the waiting time for repair, etc. If the state space is discrete, then the process is called a **chain**.

Similarly, the state index can be discrete or continuous. In most reliability engineering applications, the state index (time scale) is continuous, which means that component failure and repair times are random variables. However, cases exist where the state index is discrete. Examples include time-slotted (synchronous) communication protocol, shifts in equipment operation, etc.

Markov Processes

Markov processes are a special class of stochastic processes that uniquely determine the future behavior of the process by its present state. This means that the distributions of events (rates of occurrences) are independent of the history of the system. Furthermore, the transition rates are independent of the time at which the system arrived at the present state. Thus, the basic assumption of a Markov process is that the behavior of the system in each state is **memoryless**. The transition from the current state of the system is determined only by the present state and not by the previous state or the time at which it reached the present state. Before a transition occurs, the time spent in each state follows an exponential distribution.

In reliability engineering, these conditions are satisfied if all events (failures, repairs, switch-overs, etc.) in each state occur with constant occurrence rates (failure rate, repair rate, switch-over rate, etc.). Because the basic behavior of the process is time-independent, these processes are also called **Time Homogeneous Markov processes** or simply **Homogeneous Markov processes**. However, failure and repair rates of a component can depend upon the current state. Because of constant transition rate restriction, the Homogeneous Markov process should not be used to model the behavior of systems that are subjected to component wear-out characteristics. General stochastic processes should be used instead.

In most cases, special classes of the stochastic processes that are generalizations to the Homogeneous Markov processes are used. The corresponding models include:

- **Semi-Markov models.** Although very similar to Homogeneous Markov models, the transition times and the probabilities (distributions) depend on the time at which the system reached the present state. This means that the transition rates in a particular state depend on the time already spent in that state, but that they do not depend on the path by which the present state was reached. Thus, transition distributions can be non-exponential.
- **Non-homogeneous models.** Although very similar to Homogeneous Markov models, the transition times depend on the global system time rather than on the time at which the system reached the current state.

A non-exponential distribution (such as normal or Weibull) can be approximated as a set of exponential distributions. In this case, even the distributions are non-exponential, and homogeneous Markov models can be used. However, the results are approximate.

As noted earlier, Markov processes are classified based on state space and index space characteristics. The following table lists the characteristics of the four types of Markov processes and their corresponding model names.

State Space	Index Space	Common Model Name
Discrete	Discrete	Discrete Time Markov Chains
Discrete	Continuous	Continuous Time Markov Chains
Continuous	Discrete	Continuous State, Discrete Time Markov Processes
Continuous	Continuous	Continuous State, Continuous Time Markov Processes

Markov Model Types

In most reliability engineering applications, the state space is discrete and the index space (time scale) is continuous. Thus, **Discrete State Space, Continuous Index Space Homogenous Markov processes** are the most commonly implemented. Because the term Markov chain is generally used whenever state space is discrete, the above table refers to these models as **Continuous Time Markov Chains**. In many textbooks, these models are simply called **Continuous Markov Models**.

In addition to being an important concept in reliability analysis, Markov models find wide applications in other areas, including:

- Artificial music.
- Spread of epidemics.
- Traffic on highways.
- Occurrence of accidents.
- Growth and decay of living organisms.
- Emission of particles from radioactive sources.
- Number of people waiting in a line (queue).
- Arrival of telephone calls at a particular telephone exchange.

Markov models are the only accurate method for modeling complex situations. The complex proofs related to these models can be found in many reliability engineering handbooks and related publications.

Limitations of Homogeneous Markov Models

Homogeneous Markov models are limited by two major assumptions:

- The transitions (probabilities) of changing from one state to another are assumed to remain constant. Thus, a Markov model is used only when a constant failure rate and repair rate assumption is justified.
- The transition probabilities are determined only by the present state and not by the system's history. This means future states of the system are assumed to be independent of all but the current state of the system.

Part 2 of this Technical Brief will discuss the creation of state transition diagrams for Markov analyses.

Markov Analysis, Part 2

Creating State Transition Diagrams for Markov Analyses

State Transition Diagrams

Markov state transition diagrams are graphical representations of system states and the possible transitions between these states. They provide a visual aid to help understand Markov models. A state transition diagram can graphically represent all:

- System states and their initial conditions.
- Transitions between system states and corresponding transition rates.

In some cases, analysts represent continuous Markov models in terms of their discrete equivalents. The transition rates are replaced with equivalent transition probabilities considering that the state transition time is very small (Δt). This leads to a situation where the system can remain in the current state after time t with some probability. Thus, in this case, the probabilities of remaining in the existing state (transition rates) are also shown in the diagram.

A given system configuration is considered, at any instant in time, to exist in one of several possible states. In a single diagram, all of the operational and failure states of the system and the possible transitions between them are shown. The state transition diagram displays system states as individual nodes and transitions as either arrows or arcs.

An Example of a Single-Component System

Consider a non-repairable component with a constant failure rate (λ). The component has two states: good and failed. The states of the system are equivalent to the states of the component. Initially, assume that the component is good. The system reaches the failed state when the component fails. Once the system reaches a failed state, it will remain there forever because no events occur in the failed state. The state transition diagram of this single-component system can be represented as shown in Figure 1.

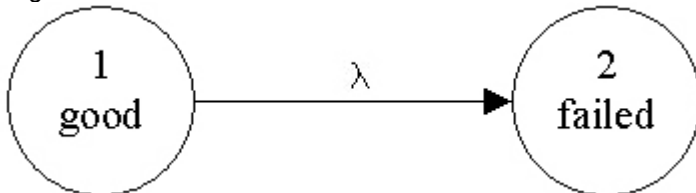


Figure 1. Single-Component, Non-Repairable System

Because state transition diagrams are more visual than mathematical matrix representations, they are much easier to interpret. A state transition diagram is similar to a flow diagram representation that would be used in system analysis. It graphically represents the various system states and the rates associated with the transitions between the system states. Because a direction is associated with a transition, a state transition diagram can be viewed as a directed graph.

Construction of State Transition Diagram

The basic steps in constructing state transition diagrams are:

1. Define the failure criteria of the system.
2. Enumerate all of the possible states of the system and classify them into good or failed states.
3. Determine the transition rates between various states and draw the state transition diagram.

An Example of a Two-Component System

Assume that there are two components in a system (labeled A and B), and that these components are in parallel. Thus, the system will function properly as long as at least one of the two components is good. Also assume that λ_1 and λ_2 are the failure rates of component A and component B respectively. Therefore, the system has a total of four states (labeled S1, S2, S3, and S4):

- **S1.** Component A is good, and Component B is good. (The system is good.)
- **S2.** Component A is good, but Component B has failed. (The system is good.)
- **S3.** Component B is good, but Component A has failed. (The system is good.)
- **S4.** Component B has failed, and Component A has failed. (The system has failed).

Of the four system states possible, only one, S4, is a failed state. The state transition diagram of this two-component system is shown in Figure 2. Because the two components in this example are assumed to be independent and non-repairable, this problem can be solved using a combinatorial model such as an RBD.

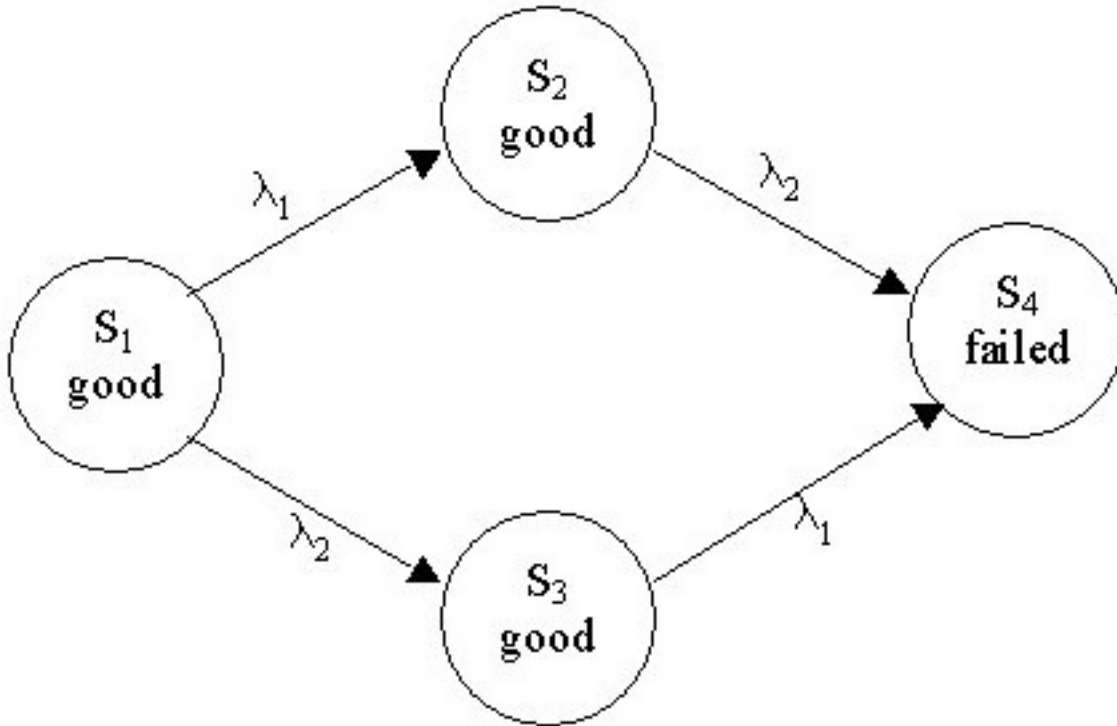


Figure 2. Two-Component, Non-Repairable System

Generally, the arrow representing the initial state is omitted from the diagram because:

- The initial state is generally where all components are in the good condition. In this example, S1 is the initial state.
- Multiple initial states can exist, such as when there are multiple phases of a mission. In these cases, all initial states are assigned probabilities that are represented by an initial state probability vector.

Now, assume that the components can be repaired as long as there is no system failure. This means that failed components can be repaired in state S2 and state S3. Also assume that μ_1 and μ_2 are the repair rates of component A and component B respectively. Figure 3 shows a state transition diagram that can represent this system. This problem cannot be solved using combinatorial models.

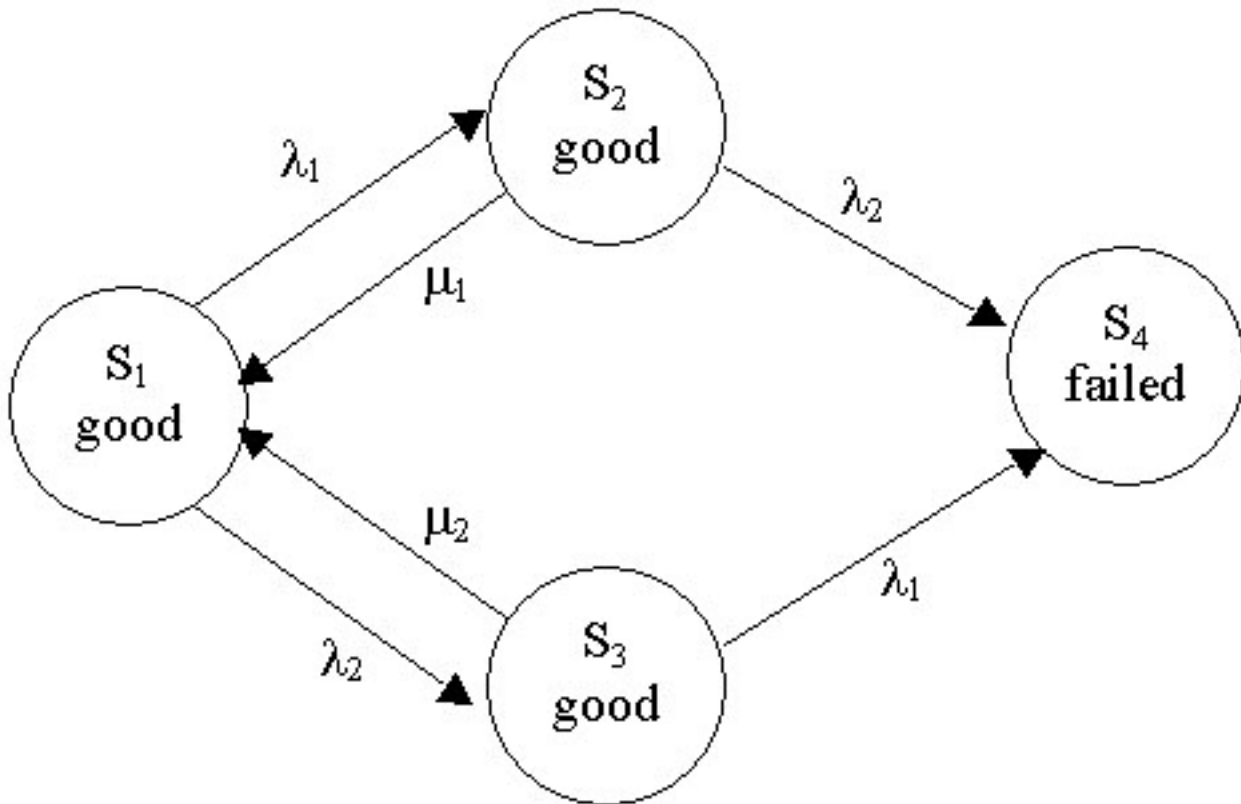
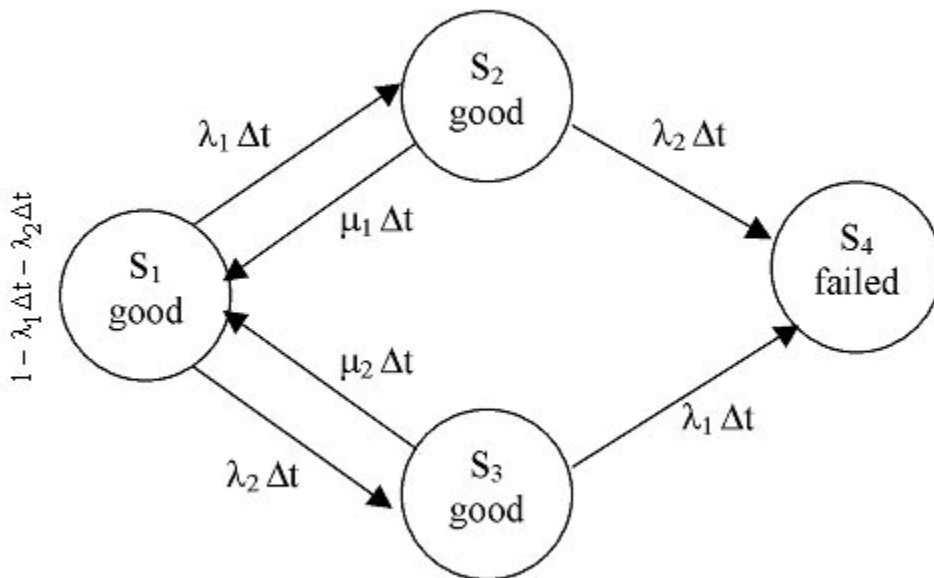


Figure 3. Two-Component, Non-Repairable System with Repairable Components

In some textbooks, the state transition diagrams of continuous models are represented using their discrete equivalents. For example, if λ is the transition rate from state i to state j , then the probability of occurrence of that transition within Δt (a small increment of t) is approximately equivalent to $\lambda \Delta t$. If there are multiple events that can occur in that state and their summation is λ , then $\lambda \Delta t$ is equivalent to the probability of transition within Δt . This shows that $1 - \lambda \Delta t$ is the probability of no transition occurring within Δt . Figure 4 shows this state transition diagram.

$$1 - \lambda_2 \Delta t - \mu_1 \Delta t$$



$$1 - \lambda_1 \Delta t - \mu_2 \Delta t$$

Figure 4. Two-Component, Non-Repairable System with Comparable Components (in Terms of Transition Probabilities)

In all of the examples presented so far, it is assumed that the system state can be expressed as the combinations of a component state. However, in some cases, the order of the events (failures, for example) is important. Suppose that each of these states has a different effect on system reliability and fail-safety. The probability of component A failing before component B

fails and the probability of component B failing before component A fails must then be known. For this example, five system states (labeled S1, S2, S3, S4, and S5) exist.

- **S1.** Component A is good, and Component B is good. (The system is good.)
- **S2.** Component A is good, but Component B has failed. (The system is good.)
- **S3.** Component B is good, but Component A has failed. (The system is good.)
- **S4.** Component A has failed, and then Component B has subsequently failed. (The system has failed in mode 1.)
- **S5.** Component B has failed, and then Component A has subsequently failed. (The system has failed in mode 2.)

Figure 5 shows a state transition diagram of this system without considering repairs. Problems considering sequence cannot be solved using combinatorial models.

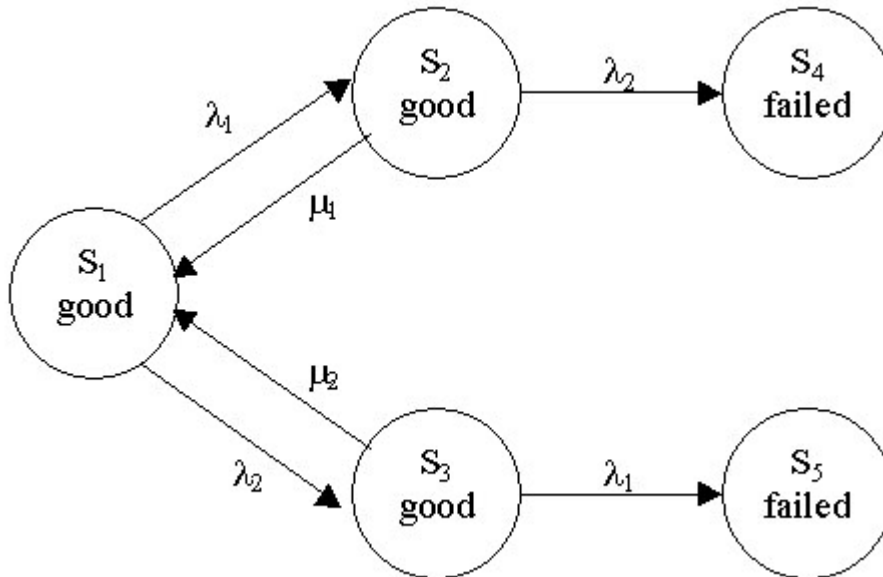


Figure 5. Two-Component System, Sequence-Dependent Failure Modes

The previous discussion shows that finding all of the system failure states may not always be simple. The following approach to constructing a state transition diagram is recommended:

1. Understand the system and the behaviors that are going to be modeled, drawing each system state in the state transition diagram.
2. Find the initial state of the system (which is generally where all components are in a good condition) and then classify each state (Good, Failed, etc.).
3. Determine all events that can occur in each state (component failures, repairs, external events such as common cause failures, etc.).
4. For each event that can occur in a state:
 - Find the state that corresponds to the event's occurrence. If this state already appears in the state transition diagram, then draw a transition from the current (initial) state to the succeeding (next) state. Otherwise, create a new state and then draw the transition.
 - Set the rate for this transition, which is the event occurrence rate (such as a failure rate or repair rate).
 - Classify the state (Good, Failed, etc.).
5. Repeat steps 3 and 4 for each state. The state transition diagram is completed when all states are visited and there are no states left to create.

After constructing the state transition diagram, adding the following information can be useful.

- Initial condition. Generally the initial condition (state probability) is 1 for the perfect state of the system (which is where this example starts, and 0 for all other states).
- Capacity. The throughput or reward of the system.

The following information is also useful for constructing state transition diagrams:

- Results from Failure Mode and Effects Analysis (FMEA) can help to identify all possible failures of a component.
- An absorbing state is a state in which no events can occur. Once a system reaches an absorbing state, it cannot visit any other state. Therefore, there are no outward transitions from this state. Generally, all absorbing states are failed states.

- Between one state and another, there can be only one transition. If multiple events make this transition, all transition rates between these two states should be added together and then this value assigned to the transition.
- Similar states are generally merged to reduce the state space and keep the state transition diagrams neat and readable. Any two states having the same transitions going out from them are treated as if they had the same set of succeeding states and corresponding transitions rates.
- All failed and absorbing states can be merged to a single state if there is no interest in analyzing individual failures, i.e., when all failed states are of the same type.
- If the sequence in which failures occur is important to identifying the type of state (Good, Failed, etc.), states should not be merged based on the combination of component failures. Otherwise, states can be merged on this basis.

[Part 1](#) of this Technical Brief discusses how Markov analyses accurately model dynamic behaviors.

Analyze Failure Data Accurately with Weibull Analysis

Handles a Broad Range of Applications, Industries, and Processes

Among all of the distributions available for reliability calculations, the Weibull distribution is the only one unique to the engineering field. Originally proposed in 1937 by Professor Waloddi Weibull (1887-1979), the Weibull distribution is one of the most widely used distributions for failure data analysis, which is also known as life data analysis because life span measurements of a component or system are analyzed.

A Swedish engineer and mathematician studying metallurgical failures, Professor Weibull pointed out that normal distributions require that initial metallurgical strengths be normally distributed, which is not necessarily the case. He noted the need for a function that could embrace a great variety of distributions, including the normal.

When delivering his hallmark American paper in 1951, *A Statistical Distribution Function of Wide Applicability*, Professor Weibull claimed that life data could select the most appropriate distribution from the broad family of Weibull distributions and then fit the parameters to provide reasonably accurate failure analysis. He used seven vastly different problems to prove that the Weibull distribution could easily be applied to a wide range of problems.

The initial reaction to the Weibull distribution was generally that it was too good to be true. However, pioneers in the field of failure data analysis began applying and improving the technique, which resulted in the U.S. Air Force recognizing its merit and funding Professor Weibull's research until 1975.

Today, Weibull analysis refers to graphically analyzing probability plots to find the distribution that best represents a set of life data for a given failure mode. Although the Weibull distribution is the leading method worldwide for examining life data to determine best-fit distributions, other distributions occasionally used for life data analysis include the exponential, lognormal, and normal. By "fitting" a statistical distribution to life data, Weibull analysis provides for making predictions about the life of the products in the population. The parameterized distribution for this representative sample is then used to estimate such important life characteristics of the product as reliability, probability of failure at a specific time, mean life for the product, and the failure rate.

Advantages of Weibull Analysis

Weibull analysis is extensively used to study mechanical, chemical, electrical, electronic, material, and human failures. The primary advantages of Weibull analysis are its ability to:

- Provide moderately accurate failure analysis and failure forecasts with extremely small data samples, making solutions possible at the earliest indications of a problem.
- Provide simple and useful graphical plots for individual failure modes that can be easily interpreted and understood, even when data inadequacies exist.
- Represent a broad range of distribution shapes so that the distribution with the best fit can be selected.
- Provide physics-of-failure clues based on the slope of the Weibull probability plot.

Although the use of the normal or lognormal distribution generally requires at least 20 failures or knowledge from prior experience, Weibull analysis works extremely well when there are as few as 2 or 3 failures, which is critical when the result of a failure involves safety or extreme costs. WeiBayes, a distribution in the Weibull family, can even be used with no failures when prior engineering knowledge is sufficient.

Weibull Probability Plots

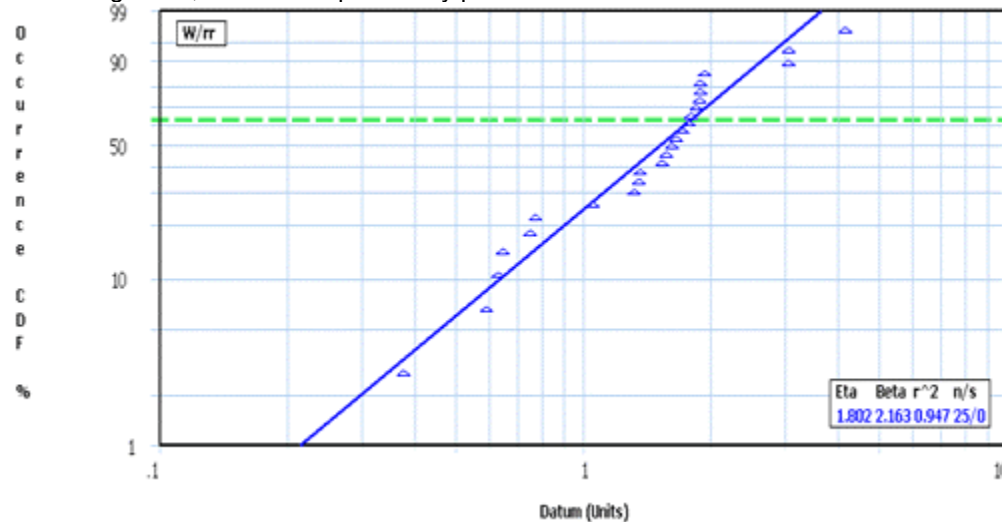
Weibull analysis studies the relationship between the life span of a component and its reliability by graphing life data for an individual failure mode on a Weibull probability plot. Weibull analysis is most often used to describe the time to failure of parts. These can be light bulbs, ball bearings, capacitors, disk drives, printers, or even people. Failure modes include cracks, fractures, deformations, or fatigue due to corrosion, excessive physical stress, high temperature, infant mortality, wearout, etc.

When plotting the time-to-failure data on a Weibull probability plot, engineers prefer using median rank regression as the parameter estimation method. Median rank regression finds the best-fit straight line by using least squares regression (curve fitting) to minimize the sum of the squared deviation (regressing X on Y). Median rank regression is considered the standard parameter estimation method because it provides the most accurate results on the majority of data sets.

Typically, the horizontal scale (X-axis) measures the component age, and the vertical scale (Y-axis) measures the cumulative percentage of the components that have failed by the failure mode under consideration.

A Weibull probability plot has a linear/nonlinear time-scale along the abscissa and another nonlinear scale for the distribution function along the ordinate. These nonlinear scales are selected in such a way that the model used for data is an appropriate one. If the scales match the data, the graph turns out to be a straight line. Because of their simplicity and usefulness, probability graphs have been used for many years in statistical analysis. However, it must be noted that the probability plotting methods to derive distribution parameters are independently and identically distributed. This is usually the case for non-repairable components and systems but may not be true with failure data from repairable systems.

In the following figure, the Weibull probability plot considers the times to failure for a unique failure mode. When a number of parts are tested under normal operating conditions, they do not all fail at the same time for the same cause. The failure times for any one cause tend to concentrate around some average, with fewer observations existing at both shorter and longer times. Because life data are distributed or spread out like this, they are said to follow a distribution. To describe the shape of a distribution, which tends to depend upon what is being studied, statistical methods are used to determine a formula. If the plotted data points fall near the straight line, the Weibull probability plot is considered reasonable.



NOTE: Although the Y-axis values are probabilities that go from 1 to 99, the distances between the tick marks on this axis are not uniform. Rather than being based on point changes, the distances between tick marks on both the Y and X axes of the Weibull probability plot are based on percentage changes. Known as a logarithmic scale, the distance from 1 to 2, which is a 100 percent increase, is the same as the distance from 2 to 4, which is another 100 percent increase. A logarithmic scale provides for apple-to-apple comparisons of several series. In addition to offering more insight into the problem, this visual representation helps to identify the distribution method that best fits a straight line to the data set.

While the above figure plots occurrences, it is very common to plot the age of components at failure. In these cases:

- The Y-axis is usually $\ln \left\{ \ln \left[\frac{1}{1 - F(t)} \right] \right\}$.
- The X-axis is $\ln(t)$.
- The Y-axis intercept is $\beta \cdot \ln(\eta)$.

Uses for Weibull Analysis

Weibull analysis has traditionally be used for analyzing failure data for:

- Development, production, and service.
- Quality control and design deficiencies.
- Maintenance planning and replacement strategies.
- Spare parts forecasting.
- Warranty analysis.
- Natural disasters (lightning strikes, storms, high winds, heavy snow, etc.).

New applications of Weibull analysis include medical research, instrument calibration, cost reduction, materials properties, and measurement analysis.

FMEAs Promote Improved Product Reliability

Anticipate Problems and Minimize Their Occurrence and Impact

Failure Modes and Effects Analysis (FMEA) is one of the most widely used and effective tools for developing quality designs, processes, and services.

NOTE: When criticality is considered, a FMEA is often times referred to as a FMECA (Failure Modes, Effects, and Criticality Analysis). In this document, the term FMEA is used in a general sense to include both FMEAs and FMECAs.

Developed during the design stage, FMEAs are procedures by which:

- Potential failure modes of a system are analyzed to determine their effects on the system.
- Potential failure modes are classified according to their severity (FMEAs) or to their severity and probability of occurrence (FMECAs).
- Actions are recommended to either eliminate or compensate for unacceptable effects.

When introduced in the late 1960s, FMEAs were used primarily to assess the safety and reliability of system components in the aerospace industry. During the late 1980s, FMEAs were applied to manufacturing and assembly processes by Ford Motor Company to improve production. Today, FMEAs are being used for the design of products and processes as well as for the design of software and services in virtually all industries. As markets continue to become more intense and competitive, FMEAs can help to ensure that new products, which consumers demand be brought to market quickly, are both highly reliable and affordable.

The principle objectives of FMEAs are to anticipate the most important design problems early in the development process and either to prevent these problems from occurring or to minimize their consequences as cost effectively as possible. In addition, FMEAs provide a formal and systematic approach for design development and actually aid in evaluating, tracking, and updating both design and development efforts. Because the FMEA is begun early in the design phase and is maintained throughout the life of the system, the FMEA becomes a diary of the design and all changes that affect system quality and reliability.

Types of FMEAs

All FMEAs focus on design and assess the impact of failure on system performance and safety. However, FMEAs are generally categorized based on whether they analyze product design or the processes involved in manufacturing and assembling the product.

- **Product FMEAs.** Examine the ways that products (typically hardware or software) can fail and affect product operation. Product FMEAs indicate what can be done to prevent potential design failures.
- **Process FMEAs.** Examine the ways that failures in manufacturing and assembly processes can affect the operation and quality of a product or service. Process FMEAs indicate what can be done to prevent potential process failures prior to the first production run.

Although FMEAs can be initiated at any system level and use either a top-down or bottom-up approach, today's products and processes tend to be complex. As a result, most FMEAs use an inductive, bottom-up approach, starting the analysis with the failure modes of the lowest level items of the system and then successively iterating through the next higher levels, ending at the system level. Regardless of the direction in which the system is analyzed, all potential failure modes are to be identified and documented on FMEA worksheets (hard copy or electronic), where they are then classified in relation to the severity of their effects.

In a very simple product FMEA, for example, a computer monitor may have a capacitor as one of its components. By looking at the design specifications, it can be determined that if the capacitor is open (failure mode), the display appears with wavy lines (failure effect). And, if the capacitor is shorted (failure mode), the monitor goes blank (failure effect). When assessing these two failure modes, the shorted capacitor would be ranked as more critical because the monitor becomes completely unusable. On the FMEA worksheet, ways in which this failure mode can either be prevented or its severity lessened would be indicated.

Approaches to FMEAs

Product and process FMEAs can be further categorized by the level on which the failure modes are to be presented.

- **Functional FMEAs.** Focus on the functions that a product, process, or service is to perform rather than on the characteristics of the specific implementation. When developing a functional FMEA, a functional block diagram is used to identify the top-level failure modes for each functional block on the diagram. For a heater, for example, two potential failure modes would be: "Heater fails to heat" and "Heater always heats." Because FMEAs are best begun during the conceptual design phase, long before specific hardware information is available, the functional approach is generally

the most practical and feasible approach by which to begin a FMEA, especially for large, complex products or processes that are more easily understood by function than by the details of their operation. When systems are very complex, the analysis for functional FMEAs generally begins at the highest system level and uses a top-down approach.

- **Interface FMEAs.** Focus on the interconnections between system elements so that the failures between them can be determined and recorded and compliance to requirements can be verified. When developing interface FMEAs, failure modes are usually developed for each interface type (electrical cabling, wires, fiber optic lines, mechanical linkages, hydraulic lines, pneumatics lines, signals, software, etc.). Beginning an interface FMEA as soon as the system interconnections are defined ensures that proper protocols are used and that all interconnections are compliant with design requirements.
- **Detailed FMEAs.** Focus on the characteristics of specific implementations to ensure that designs comply with requirements for failures that can cause loss of end-item function, single-point failures, and fault detection and isolation. Once individual items of a system (piece-parts, software routines, or process steps) are uniquely identified in the later design and development stages, FMEAs can assess the failure causes and effects of failure modes on the lowest level system items. Detailed FMEAs for hardware, commonly referred to as piece-part FMEAs, are the most common FMEA applications. They generally begin at the lowest piece-part level and use a bottom-up approach to check design verification, compliance, and validation.

Variations in design complexity and data availability will dictate the analysis approach to be used. Some cases may require that part of the analysis be performed at the functional level and other portions at the interface and detailed levels. In other cases, initial requirements may be for a functional FMEA that is to later progress to an interface FMEA, and then finally progress to a detailed FMEA. Thus, FMEAs completed for more complex systems often include worksheets that employ all three approaches to FMEA development.

Use Relx Weibull to Minimize Downtime or Overall System Cost

Optimize Replacement Strategies Based on Block of Time or Age Maintenance Policies

Block of Time Replacement Strategy

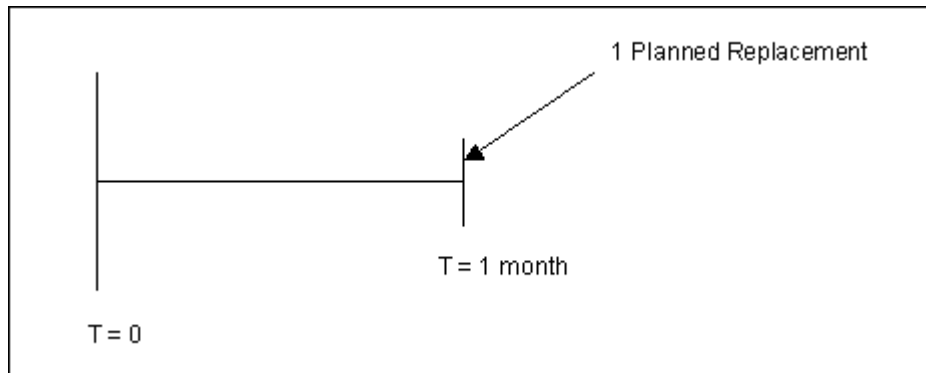
With this strategy, components are replaced at a predetermined, non-moving interval of time. For instance, a component might be replaced on the first of every month, regardless as to whether or not it has been recently replaced due to failure. Although this strategy is very simple to implement, it results in more replacements.

The block of time replacement strategy is optimized using a renewal function, which is an exact solution. It varies the block of time starting at a user-defined minimum block and increases the block of time to the user-defined maximum block. If cost is being optimized, the strategy then picks the minimum total cost.

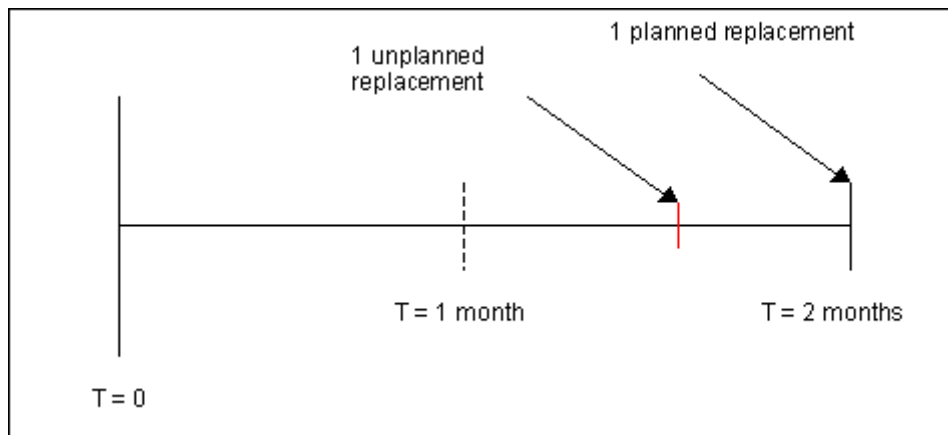
Example

Let us assume that the optimal replacement strategy for an industrial hydraulic valve is to be determined using a block of time replacement strategy that minimizes the costs incurred from replacements. Let us also assume that unplanned replacements are 10 times more expensive than planned replacements, and that the block of time is to be at least 1 month but no more than 12 months.

At one month, the Relx Weibull module will have determined that the component has not failed, as pictured below:



Let us assume that the Relx Weibull module then increases the time to 2 months (although in practice it would use a much smaller increment than this). Even though an unplanned failure occurs between the first and second months, the planned replacement still occurs at two months as shown in the next figure:



After a number of iterations, the Relex Weibull software module would calculate:

$$\frac{[\text{Cost of Planned Replacement}] * [\text{Cost of Unplanned Replacement}] * [\text{Renewal Function}]}{\text{Time}}$$

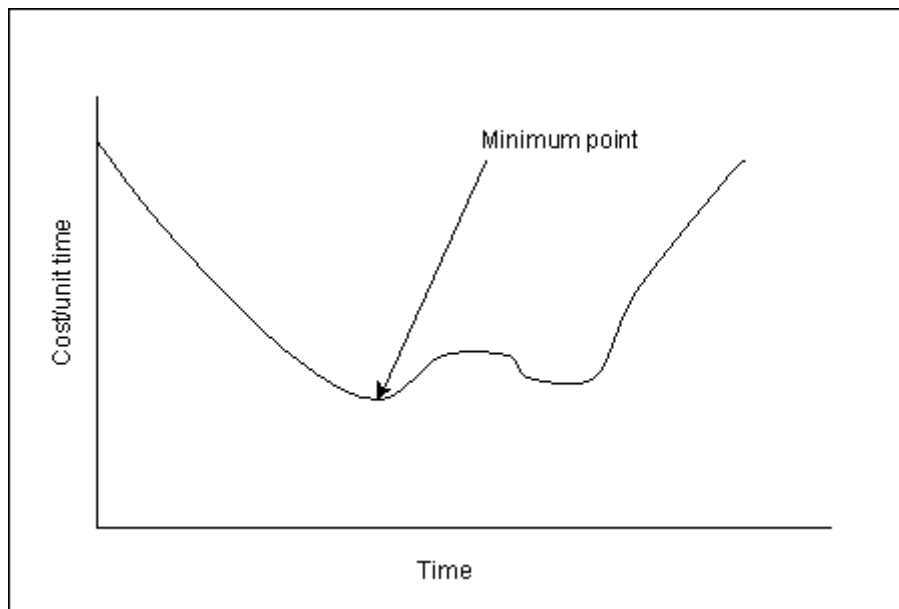
Age Replacement Strategy

With this strategy, components are replaced at a given age on a sliding time scale. For instance, a component might be replaced one month after its last replacement. Although this strategy is more difficult to implement, it results in fewer replacements occurring at more ideal times.

The age replacement strategy is optimized using a residual life time function, which is an approximation.

Example

An example failure graph is shown below for a case where the component is replaced exactly one month after the previous failure. Calculations are performed similarly to the block of time calculations.



Conclusion

Within the Relex Weibull module, the Optimal Replacement Wizard makes determining the optimal replacement strategy easy. You simply select a failure distribution and then complete the Optimal Replacement Data dialog box:

Optimal Replacement Data

Replacement Objective: _____ Replacement Time Interval: _____

Minimize Cost Minimum:

Minimize Downtime Maximum:

Replace Based On: _____ Cost of Replacement: _____

Failure or Age Planned:

Failure or Block of Time Unplanned:

Maximum Number of Failures in Replacement Interval: _____

< Back Next > Cancel Help

1. Select your replacement objective.
2. Specify the minimum and maximum value for the time interval.
3. Select the Age or Block of Time replacement strategy.
4. If the Block of Time strategy is selected, indicate the maximum number of replacements that occur due to failure in the replacement time interval in the field that becomes available.

Using Fault Trees to Identify Potential Faults in Critical Systems

Visualize the Events that Lead to Component Failure

Fault Tree Analysis (FTA) is well recognized worldwide as an important tool for evaluating safety and reliability in system design, development, and operation. For more than 40 years, FTA has been used in the aerospace, nuclear, and transportation industries to translate the failure behavior of a system into a visual diagram that displays system relationships and root cause failure paths. A fault tree provides a concise, visual representation of the various combinations of possible occurrences within a system that can result in a predefined and undesirable event. FTA is most often used for:

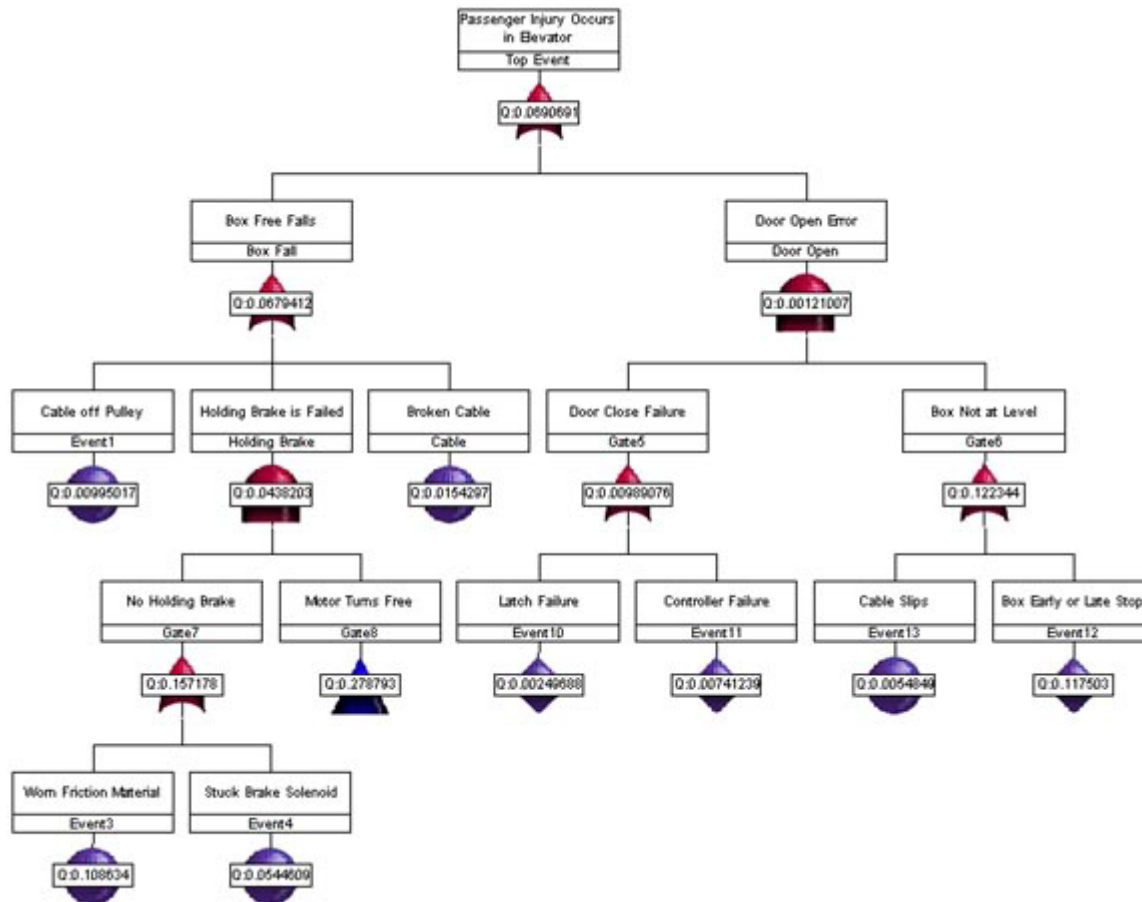
- Identifying safety critical components.
- Verifying product requirements.
- Certifying product reliability.
- Assessing product risk.
- Investigating accidents/incidents.
- Evaluating design changes.
- Displaying the causes and consequences of events.
- Identifying common-cause failures.

FTA is a deductive analysis method that begins with a general conclusion (a system-level undesirable event) and then attempts to determine the specific causes of this conclusion. Based on a set of rules and logic symbols from probability theory and Boolean algebra, FTA uses a top-down approach to generate a logic model that provides for both qualitative and quantitative evaluation of system reliability.

The undesirable event at the system level is referred to as the *top event*. It generally represents a system failure mode or hazard for which predicted availability data is required. The lower level events in each branch of a fault tree are referred to as *basic events*. They represent hardware, software, and human failures for which the probability of failure is given based on historical data. Basic events are linked via logic symbols (gates) to one or more undesirable top events.

Computerized FTA

Small fault trees have fewer than 100 events, medium fault trees have from 100 to 1,000 events, and large fault trees have more than 1,000 events! Today, computerized FTA can be used to analyze very complex systems as well as very complex relationships between hardware, software, and humans. Using good FTA software, you can cut, copy, paste, rearrange, and delete events and gates to various fault tree branches to quickly and easily compare different hardware configurations. An example of a computer-generated fault tree follows.*



* Generated in Relex Fault Tree. Click the image to view a full-size version.

In the above figure, "Passenger Injury Occurs in Elevator" is defined as the top event. The reasons why passenger injury in an elevator could occur have been determined to be either that the box free falls or that the door is open at an inappropriate time. After determining all possible causes for each event identified, the events and gates for connecting them to higher-level events are added to the fault tree. Any faults that can be further developed to determine causes are then added as lower-level events and connected by the appropriate gates.

The lowest-level events that terminate fault tree paths are called *basic events* or *primary events*. They are either component-level events that cannot be further resolved or external events. For example, in the first level of possible events for the free fall of the box, "Cable off Pulley" and "Broken Cable" are basic events. Because these events are primary faults, they are not developed any further in the fault tree.

Fault Tree Construction

To construct a useful fault tree, the analyst must fully understand the system as an integrated interaction of subsystems. In addition to having a logical mind and the ability to visualize the logic structure and interaction of a system and its subsystems, the analyst must have knowledge of the dependencies between the components, their reliability parameters, and the conditions that determine the components that are considered to have failed. Thus, good analysts are generally experts in mechanical, structural, electrical, and control systems and also have an understanding of human interactions, procedural implications, and even chemical interactions.

The most common errors in constructing fault trees include:

- Using too wide of a scope for the top event, which results in a large, complex, and unfocused fault tree.
- Using inconsistent nomenclature for the same events, which prevents you from finding events that occur in multiple branches of the fault tree.
- Using the same nomenclature for similar but different components, thereby identifying the same failure for several scenarios when these failures are actually caused by different components.
- Breaking the fault tree into branches by electrical, mechanical, and structural subsystems, thereby failing to take the interface and integration of the system into account.

Top Event Definition

Because the top event sets the tone for the series of questions that are considered when constructing the fault tree, the analyst should use the system definition to construct a clear and concise top event. If a top event is vaguely stated, the fault tree is likely to be large, complex, and unfocused. To generate a useful fault tree, the top event must be precisely stated and be narrow in

scope. Specifying the specific mission phase or portion of the mission to which a top event applies in the description of the top event often helps to generate a very concise fault tree.

Event Nomenclature

During fault tree creation, consistently applying the appropriate nomenclature to events is critical to identifying the same event in multiple fault tree branches. If, for example, you give an event a different name in another branch of the fault tree, cutset analysis, which is described in the "Fault Tree Analysis" section, identifies multiple events leading to different failures (rather than the same event leading to different failures). If you do not realize that nomenclature errors exist, you may not recognize an event as a major contributor to the top event and thereby fail to recommend improvements or controls for it. Similarly, when two identical components are installed in different locations within a system, you must be sure to identify that they are physically different components by using reference designators in the nomenclature. Otherwise, cutset analysis identifies how the same component failure contributes to several scenarios when the failures are actually caused by different components.

Branch Arrangement

Because engineering groups so often function autonomously, fitting each piece of hardware together in a system tends to be an afterthought. Organizations that regularly categorize work by engineering disciplines tend to arrange the branches of a fault tree by subsystems. However, such an arrangement limits FTA to considering only component failures. When engineering groups fail to properly coordinate and implement a design as a team, interfaces and interactions are most often the areas in which the system breaks down. When fault tree branches are arranged by subsystems, these areas are never even addressed.

When scenarios that lead to the top event are used to arrange fault tree branches, the analyst can place faults under the cause for a component failure. Causes can include not only hardware failures but also interface and integration problems due to design flaws, software, human errors, operation and maintenance errors, and environmental influences on the system. Fault trees arranged by scenarios often uncover complex relationships and interactions of systems, components, and actions that are believed to be unrelated. For example, such an FTA can reveal a single-point component failure that can fail two supposedly redundant or independent systems.

Fault Tree Analysis

After properly identifying all failures, events, and conditions that can lead to the occurrence of the top event, you can compute the probability of the top event and measure the relative impact of a design fix. The traditional analysis process is to generate the system minimal cut sets, apply the basic event probabilistic data, and then determine the probability of the top event.

The qualitative analysis of fault trees is based on determining the minimal cutsets for the top event. Cutsets identify the sets of events that cause the top event to occur. A cutset can be a single-point failure or event or can be a set of many events. Different cutsets can include different combinations of the same event. A minimal cutset is the smallest group of events that cause the top event to occur. In large trees, the events that cause the top event to occur are often buried deep within the system and are not easily discovered without performing cutset analysis.

The basic events that belong to a cutset provide such information as single-point failures and the relative contributions of each cutset. Generally, the cutsets that have the highest probability of occurrence are the ones that have the fewest number of events. Thus, the minimal cutset information obtained during qualitative analysis can be used for computing the unavailability and unreliability values of the system during quantitative analysis. (Unavailable and unreliability values are calculated by FTA because fault trees are organized around system failures rather than system successes.) For quantitative analysis, reliability and maintainability information such as failure probability or repair rate is used to determine or quantify the probability of occurrence of the top event.

Conclusion

Because FTA is an event-oriented analysis, it can identify more possible failure causes than structure-oriented FMEAs (Failure Modes and Effects Analysis) and RBDs (Reliability Block Diagrams), which allow only hardware failure considerations. When performed correctly, FTA often identifies system problems that other design and analytical methods would overlook.