Beauty And The Beast – Use And Abuse Of The Fault Tree As A Tool

R. Allen Long: Hernandez Engineering, Inc., Huntsville, Alabama

Keywords: fault tree, cutsets, interfaces, software

Abstract

Fault Tree Analysis (FTA) has become a popular tool for use in the Space Industry for the System Safety Engineer. The fault tree is used for everything from tracking hazard reports to investigating accidents, as well as presentations to management. Yet, experience in the space industry has shown the fault tree is used most often for purposes other than its original intent, namely for evaluating inappropriate behavior in complex systems

This paper describes proper application and common misapplications of the fault tree as a tool when evaluating inappropriate behavior in complex systems. The paper addresses common misconceptions and pitfalls about FTA such as tracking only failures, and the belief that Failure Modes and Effects Analysis (FMEA) can be used in lieu of the fault tree.

Introduction

FTA in the space industry is rarely used to its full potential. Moreover, the true beauty of the fault tree has been overlooked more often than not. FTA is perhaps the most misunderstood methodology in wide use today. Yet, FTA is one of the most beautiful of tools at an engineer's disposal. Although the concept and symbols are relatively simple, proper development of a fault tree is anything but simple. There are many myths and misconceptions surrounding this elegant tool.

A Most Beautiful and Versatile Tool For Modeling Complex Systems

There is no one right way to develop a fault tree. However, there are many wrong ways and even more hard ways to perform a fault tree. Two of the most important concepts to performing a useful FTA (and maintaining your sanity) are:

Keep Your Eye On The Prize: FTA can be developed as far down as the analyst wishes to go. Often the fault tree takes on a life of its own and drags the analyst down to the subatomic level. An easy trap to fall in is developing the tree for development's sake; thus, losing the end game. You wind up with a great tree that has no purpose. Remember the prize is finding areas for improvement on a system for which you can provide reasonable design controls. The paradox here is that the more adept you become at performing FTA, the more likely you are to develop the tree to a lower level than necessary. Go with the flow, but don't get carried away by it!

A well executed FTA goes beyond the FMEA in identifying ways to cause the Top Undesired Event. FTA can identify design, manufacturing, processing, and handling "faults" in addition to component failures.

FTA development can go far beyond a component failure if required. Skill is required not only in defining the appropriately framed Top Undesired Event, but in determining the needs of the particular customer or level of management as to how much detail is needed. Management may only need an FTA to a relatively top level. However, a chemist may need the tree to go down to "Brownian motion" in explaining a particular problem. Herein lie both the beauty and the beast of FTA.

Know Your Audience and Work the Crowd: On one hand, when performed correctly and in cooperation with the target audience, FTA can be taken to the exact level of detail needed. On the other hand, the FTA can be developed into an unwieldy beast far beyond details that are not useful for your customer. Moreover, the type of logic used (i.e., "Failures" vs. "Scenarios") can turn a large detailed tree into nothing more than a fancy FMEA or a breakout of systems and subsystems. More often than not, fault tree analysts ignore possible interactions and fail to identify scenarios that do not neatly fall within "Electrical," "Mechanical," "Structural" or similar failures. Such a tree is rarely useful, and will fail to identify many interactions that could cause the Top Undesired Event (more on this subject later).

However, I caveat this with the following: FTA can be tailored to the needs of a particular project, system engineering group, investigation team, or management organization. There have been many instances in which I have found the target audience did not want a fault tree in its purest terms. The beauty is that FTA can be adapted beyond the "FTA Purest" definition to aid many organizations in identifying solutions to problems ranging including: process control, hardware failures, interactions of personnel with machinery and processes, and inherent functionality, reliability, or safety of a system's design.

Sometimes the most difficult part of a fault tree is convincing the target audience that they need a fault tree that is relatively pure. Since the customer is always right, the analyst will often be forced against his/her better judgment to tailor the fault tree to the customer's wishes. Much to my dismay, some of these "customized" fault trees have provided me with the most positive customer feedback!

Wow! I Didn't Know the System was Going to Do That! (Or, Why Perform FTA vs. Other Analytical Methods?)

Performed correctly, FTA often identifies problems with a system other design and analytical methods may have overlooked. Complex relationships and interaction of systems, components, and actions thought to be unrelated can be discovered. On a complex system, a properly developed FTA almost always finds at least one "I didn't know the system would do that!" Call this a "Eureka." The most common "Eureka" is a common cross-link or single point failure which could fail two (supposedly) redundant or independent systems.

Eurekas are easiest found during the design of a system. You are much less likely to find a Eureka during an accident or incident investigation – although it does happen. This is especially true when the hardware is lost in space or has been consumed in a fire on a test stand.

Can You Say Interfaces? One of the sad truths in engineering today is that "integration" has been reduced to a buzzword. Lots of lip service is given to integration or the study of the interfaces. However, engineering groups are largely compartmentalized only "interfacing" with each other after they have completed their respective part of a project. Making the pieces fit together is an afterthought – usually addressed too late in the design process to provide a truly integrated product or process. Each engineering discipline basically bolts on its piece of hardware to the project. It is this "cross-strapping" of systems where the fault tree can pay for your effort. This is the most likely place where problems have not been adequately addressed nor discovered by the various designers. This is where other analysis methodologies and engineering disciplines fail and where the system safety engineer can convince other engineering groups that we are a legitimate "engineering" discipline.

Develops Strong Minds and Bodies (of System Engineering Talent) One of the greatest benefits of FTA may come in the form of system engineers who are much more knowledgeable of the system. It is not unusual for a fault tree analyst to understand the system and the interaction/relationship of its subsystems better than the design or system engineers with whom the fault tree analyst is working. We talk about integration in aerospace but often do not properly train our engineers nor properly analyze the system parts in a manner which will truly ensure proper integration. FTA performed by an experienced fault tree analyst together with a systems engineer provides the project with a systems engineer who is much more knowledgeable than when the engineer started.

Simple Rules of Thumb In Performing FTA

How far is too far? Conquering the beast requires the analyst to develop the tree to a manageable level of detail. But, how do you determine how far is too far?

A good rule of thumb is to develop the fault tree down to the level at which you can exert reasonable control. For example, if you are analyzing a safe-&-arm system that involves a programmable controller, you probably do not have to go into the controller as part of your analysis. Assume the controller can malfunction and analyze the rest of the system to ensure there are controls and/or inhibits to prevent inadvertent firing of the circuit if the controller malfunctions.

If your system deals with "line replaceable units" (LRU's) you probably should stop at the LRU level.

There may be times when you are analyzing the black box itself. In these cases, you take the fault tree development to the integrated circuit (i.e., the "chip" level). Chances are you can recommend additional chips, additional or different wiring or other components in the black box. But, you will almost never be able to change the internal design of the chip. Don't even go there!

FMEA vs. FTA. Or, "Push That Button and We're All Gonna Die!"

Often FMEA and FTA are mentioned in the same sentence. A common fallacy is to assume that use of the FMEA can supplant the FTA. It is not unusual for the aerospace community to state that every FTA basic event must match up with a listing in an FMEA. Remember, FMEA's are based on the following assumptions:

1. Inputs into the component are correct (e.g., high pressure is not introduced into a low-pressure system).

2. Assume that Operational Environment is proper for the component (e.g., it is assumed that a sensor in a corrosive atmosphere is properly designed for that atmosphere. The atmosphere does not contribute to the sensor failure in the FMEA).

3. Only single point failures are addressed. Redundancy is only addressed in terms of "Like-Redundancy" of the same component.

4. Human error is groundruled out. Assembly, maintenance, and operational errors are not addressed in the FMEA methodology. Lifting operations is a good example where FMEA can only address a very few of the potential hazards since most hazards are not due to component failure.

5. The overall design is assumed to be correct. The only exception of this is identifying single point component failures that can cause critical or catastrophic events. The component failures are usually addressed in terms of the expected failure rate or life of the component (i.e., common cause failures, inputs, and environmental factors are not taken into account). Even a component failure due to damage from an improper input, handling, or installation damage/errors is not addressed.

To illustrate this concept let's look at a valve in Figure 1, which is commanded open by a programmable controller due to a change in pressure. For this example, let's assume the valve is a remote-operated pneumatic valve in a gaseous oxygen (GOX) system. This pneumatic valve is driven open when air is supplied and spring-driven closed when air is removed. Such a valve would typically involve a solenoid valve which when electricity is supplied, the solenoid moves and diverts air to the pneumatic valve to drive the pneumatic valve open. When power is removed from the solenoid it closes via a spring. Therefore, loss of electrical power would cause the valve to close. A pressure transducer or switch signals the programmable controller. The programmable controller sends the electrical signal to the pneumatic valve solenoid depending upon the pressure transducer signal.

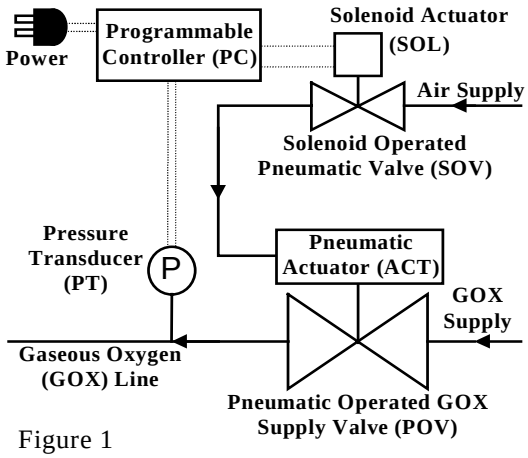

Figure 1

In our example, we are concerned with the valve inadvertently closing. The following can cause this:

1. Improperly commanded closed by the programmable controller: controller failure, faulty software, temporary interruption of programmable controller function due to an electrical transient, or improper start-up or initialization by the operator.

2. Loss of power at the solenoid due to: 1) cable failure/signal interruption between the controller and the solenoid, 2) physical damage to the wire/cable harness from

vibration, operating environment, damage from adjacent operations/equipment, short or fault, overcurrent, 3) loss of facility electrical power, or 4) contacts unable to properly conduct power due to corrosion.

3. Solenoid Failure caused by: coil failure, contact failure (inability to move/switch), or mechanism failure/seizure.

4. Pressure Transducer fails to provide proper signal.

5. Loss of air supply. The controller sends the proper signal and the solenoid has properly activated, but there is no air to drive the pneumatic valve open. This could be caused by: loss of air supply at the source (tank or gas plant), venting of the air supply via relief valve (failure or system overpressurization), or damaged/broken, crimped or clogged airline.

Several of these causes would be identified in the FMEA. However, many would not. Even in those component failures that are addressed in an FMEA, many of the underlying causes would not be identified. For instance, the pressure switch might not be rated for the operating environment. It therefore fails under normal operating pressure because it was under-designed. As a single point failure, the FMEA might recommend a redundant switch. However, if the basic design flaw is not identified, the redundant switch would also fail under normal operating pressures. The switch might also be overpressurized due to a flaw in the system design *upstream* of the switch. An upstream regulator might be improperly set (i.e., it doesn't fail--the set-point is too high). In such a case, is there a relief valve in the system to prevent overpressurization of the system? The relief valve could simply fail; or it could fail to relieve because a bird's nest or mud dauber hive is blocking the orifice; or, the relief valve's set point is too high. The same valve installation/calibration/maintenance schedule or program which allows an improperly set or maintained regulator could also allow the relief valve to be improperly set or maintained!

This is where the fault tree struts its stuff. Events in a fault tree are not always based upon a component failure. The fault tree can find problems within a system due to operator error, design flaws, or undesirable interactions in which combinations of "benign" failures or normally expected operating conditions cause the top undesired event. Interfaces and interactions are most often the area in which a system breaks down, malfunctions, or causes the Top Undesired Event to occur. For instance, a sensor reading anomaly due to a sensor failure itself is rarely the problem. The chain of events set in motion by an erroneous sensor is the potential problem. This occurs at or across a system interface, and, for purposes of this discussion, what if the sensors are redundant? Is not the problem solved?--not necessarily. An FMEA would most likely identify the sensor as redundant and therefore not a problem. What if both sensors are defective due to a manufacturing lot problem? An erroneous reading might not be due to a sensor failure but due to a problem with the software or data interpretation. Both sensors might be operating exactly as designed but data is lost between the sensors and the control system, The control system may react improperly to a proper incoming signal. Use of an improper sensor for the environment could cause both sensors to fail due to degradation.

Totally redundant systems designed by one group are usually controlled by systems designed by a different group. It is at this interface that the problem usually arises. The control system group may actually tie both redundant systems into a single failure point. The failure point may in fact not be a failure at all but an unexpected consequence of a normally expected event. One rocket motor manufacturer used a device called a Vacu-Lift, which would basically grab onto and lift an empty motor case using a vacuum system. The redundant vacuum arrangement in the Vacu-Lift was truly a thing of beauty. However, the two totally redundant vacuum systems were tied into a single venting solenoid. Activation of this solenoid (via a single button) caused both sides of the vacuum system to simultaneously vent (or draw atmosphere). Thus, the elegantly designed and executed redundant vacuum system was completely undermined by a flawed execution of the control system. This is a good example of two engineering groups failing to properly coordinate and implement the design as a team (i.e., in this case, one engineer in charge of the vacuum and mechanical systems and another engineer in charge of the control system). Because of the single solenoid, dropping the case due to venting the vacuum through the solenoid could occur

because of: 1) solenoid Failure, 2) vent button failure, and/or 3) operator accidentally pushing the button. In this case, pressing a single button could have truly caused personnel to die.

Another area the FMEA does not address is damage to a system. Let's use the common automobile for a couple of examples to illustrate FTA vs. FMEA: Let's say we are investigating an accident in which you have narrowed the cause down to a braking problem. The FMEA will assume that the problem is a Braking System Failure. However, many likely causes will not be covered using the FMEA approach. Examples could include: slippery roads, failure of the driver to brake in time (or not at all), bald tires, improper brake fluid (wrong type), wet or oily brake pads. Remember that most FMEA assume inputs to the system are correct.

There are other issues that might or might not be addressed using the FMEA depending upon the groundrules used by the analyst such as insufficient brake fluid or worn brake pads. Technically, the pads have not failed. They just may be less effective than new pads, depending upon the analyst's point of view.

Another example using the automobile could be if the engine fails to run. Note that the term "Engine Failure" is not used here. The engine can fail to run for dozens of reasons not associated with an engine component failure. The engine could seize due to: 1) an insufficient amount or the quality of oil (the owner has not put oil in the car for some time or the owner put in the wrong oil), 2) the owner went too long between oil and filter changes, or 3) the oil filter was not properly tightened and a leak occurs between the filter and engine. Another reason could be ignition switch failure. Depending upon the FMEA analyst, this might be considered an input or not part of the engine proper and he could ground-rule it out.

The engine may stall due to a fuel problem. This might be caused by: 1) a clogged filter (may or may not be considered a "Failure"), 2) a malfunctioning gas pump (also likely not to be considered as part of the engine proper), 3) water or other contaminate in the gas (such as sugar in the gas tank from your teenager's ex-boyfriend or girlfriend), 4) clogged or crimped gas line/vapor lock, 5) no gas in the gas tank or fuel line leak.

The Three Biggest Myth-Conceptions Uncovered

Myth Number 1. -- "Oh, I can do FTA!" Or, Everyone is an Expert in FTA: FTA looks deceptively simple yet is rarely done correctly by the FTA newcomer. Lots of FTA's are produced but few are actually of any significant value (other than *wowing* others with an impressive graphic that falsely indicates progress was made).

Fault tree analysts must have a logical mind and the ability to VISUALIZE the logic structure and interaction(s) of a system and its subsystems. The analyst must understand the system as an ***integrated interaction*** of subsystems.

A fault tree analyst must also have sufficient understanding of the details of a wide variety of systems and subsystems. The analyst must be able to understand everything from chemical interactions, electrical subsystems, mechanical, structural and control systems, as well as human interactions and procedural implications.

One of the problems we have in aerospace integration is that we have not developed systems experts knowledgeable in multiple technical areas. Most disciplines tend to see only their own tiny areas of expertise. Few engineers and other highly technical personnel can divorce themselves from their specific areas to look at the larger picture. Even fewer are willing to learn technical details of other disciplines and how those details affect the larger picture. The interactions and interfaces are often dealt with as necessary inconveniences to getting their particular part connected to the overall system; where-as in reality, the interfaces and interactions are usually the whole point of the project.

Unfortunately, this has lead to a plethora of FTA's that have little or no technical value. This being the case, many organizations have fallen into the trap of performing FTA simply to satisfy a contractual or management requirement. Therefore, we see poor FTA leading to apathy toward fault trees and hence, poorer FTA. The true beauty and full potential benefits of FTA have thus been lost on many organizations.

Myth Number 2. -- "Any Fault Tree Software Will Work." (The so-called good programs are too expensive.): Don't save software dollars and waste engineering megabucks ***and***, do not confuse

FTA software graphics with CAD or presentation graphics packages.

Good fault tree software ***is*** expensive. No program is good at everything. Programs with poor input interfaces or unwieldy output processing will soak up so much of an analyst's time that the most expensive of programs could have been purchased. Unfortunately in today's budget arena, engineering resources are often paid for out of a different pot of money than is the software. Therefore, wasteful use of engineering time occurs more than management would admit. A good FTA software package in the hands of a seasoned analyst can save the company or government literally hundreds of thousands of dollars in a single year. Compared with some programs, one good analyst can do the work of five or six engineers using an inferior program.

Inexpensive programs I have tested usually are not very useful for complex systems or large trees. Often, additional site licenses and other "ups & extras" bring programs up to the same cost level as the more expensive varieties. Expect to pay around $15,000 - $20,000 for a good FTA program with several site licenses. The cost will be more than made up if your organization is serious about FTA. As large as the cost may seem, the savings in engineering hours can often be recovered on a single project.

Even if your organization is not going to perform quantitative analysis, a good FTA program is more than a fancy PowerPoint® or AutoCAD®. Good FTA software utilizes a relational database structure in referencing logic structure. This allows developing trees to compare different hardware configurations using the same basic components and events quickly and easily. Pruning, re-arranging, copying and replicating logic is very fast. On large trees, replicating "like" logic from one large branch to another takes only a few minutes. The rearranging of the tree to fit on the printout is done automatically. Such input and rearranging of the entire tree to print out can take many hours on poor FTA programs or programs that are strictly graphically oriented. Such programs require manually cutting, copying, pasting and rearranging all affected branches of a tree. It is one thing if you can just add new events to the end or bottom of a tree. If you have to add logic and additional events in the middle of a large tree, manual rearrangement can be a hair pulling exercise.

Good FTA software can be even more critical for those organizations who support accident investigations with FTA. Upper management is almost always involved in such investigations. The "Big Guns" want iterations of the Fault Tree fast! It is difficult for an organization to blame their software on why they cannot get (sometimes relatively minor) Fault Tree changes out quickly. This can be especially embarrassing (even contract limiting) if management or the customer has previously utilized other companies or organizations to perform FTA using state-of-the-art software! It has also been my experience, that a failure investigation can often be used to an organization's benefit in providing the stimulus for management to provide funding for an expensive FTA program.

Also, a good program will properly perform cutset analysis and minimize the cutsets. This is a valuable function even if you are not going to quantify the result. I CAN NOT STRESS THIS ENOUGH! Unfortunately, I do not have sufficient room to explain minimal cutsets in this paper. However, cutsets and cutset analysis are discussed in a section below. For purposes of this paper, "cutset" will always refer to a "minimal cutset" (All good programs automatically compute the minimal cutsets).

Myth Number 3. -- "Anyone With Fault Tree Software Can Develop Effective FTA": Give me a Stradivarius violin and I will play like Itzak Perlman or, for you jazz fans, Jean-Luc Ponty. Perhaps a more appropriate example would be to give me a good music composition software package. After a couple of weeks to learn the software, I will compose every bit as good as Beethoven or Bach…you think? – Enough said!

Building the Perfect Beast -- Tips, Tricks, and Pitfalls of Performing FTA

Define the Top Undesired Event as Precisely and Narrowly as Possible: Even though we've all heard that the Top Undesired Event should be properly, narrowly, and precisely phrased, this fact is blown off by many fault tree analysts. Usually, the wider the scope of the Top Undesired Event, the less useful the tree. Although the scope of the tree can be too narrow, this is rarely the case.

There was a Solid Rocket Booster nozzle anomaly on at least one of the shuttle flights. This anomaly was very specific in that it involved erosion in a geometric pattern. Initially, the team developing the fault tree began the tree with a Top Undesired Event of "Nozzle Anomaly." Neither the erosion, nor the geometric pattern was mentioned. Consequently the investigation team began floundering because the tree was too broad in scope. There could be hundreds of nozzle anomalies that had nothing to do with the erosion. There could even be dozens of anomalies due to erosion that would never cause a geometric pattern. It was not until a seasoned fault tree analyst (a non-team member) suggested that the Top Undesired Event be narrowed to include erosion and the geometric pattern that the tree and the team began to focus in on the actual problem.

Divide Your Tree Into Branches Dealing With Scenarios That Can Lead To The Top Undesired Event: Do not define the tree branches in terms of failures. Failures belong near the bottom of a branch, such as under what would cause a component to fail to function. (Failure is usually only one of several reasons something fails to function properly). Immediately breaking the tree into subsystems such as "Electrical" or "Mechanical" failures is one of the most heinous mistakes in creating a fault tree. It is also the most commonly used method of developing a fault tree. In fact this method is almost an institution among many organizations that are acquainted with fault trees and therefore believe they too are fault tree analysts. By breaking a fault tree into such sections, the analyst is led down the path of "Failures." You might as well perform a FMEA. Furthermore, interfaces will never be addressed in such a tree. Therefore the item(s) which need the most scrutiny will most certainly be omitted! The very reason many organizations fall into this trap is because this allows organizations to easily pigeonhole categories by engineering disciplines. One of the most compelling reasons to perform FTA in the first place is to ensure the interfaces are not "botched" up or will not cause problems due to poor integration.

Let's go back to our pneumatic valve example in Figure 1. If we list the Top Undesired Event as "Pneumatic Valve Failure" and list the main branches as: 1) Pneumatic Valve Mechanical Failure, 2) Pneumatic Valve Electrical Failure, and 3) Mechanical Valve Structural Failure, your tree will be very short. The tree will be a little bigger if you consider the solenoid valve as integral to the pneumatic valve (as is the case in some solenoid operated valves packages). Using "Pneumatic Valve Failure" as the Top Undesired Event would not identify electrical or command problems with the programmable controller or pressure transducer. Nor would it identify loss of electrical power, or loss of air supply to the solenoid valve. If you were to consider the solenoid valve as a separate package from the pneumatic valve, you would likely not identify the solenoid valve at all as part of the pneumatic valve failure.

Chances are, you are actually interested in what would prevent gaseous oxygen from flowing through the pneumatic valve. So, in this example, a more appropriate Top Undesired Event would be "GOX Fails to Flow Through Pneumatic Valve." Figure 2 shows what such a tree, organized into scenarios instead of "Failures" might look like.

Remember, in dealing with components, failures are only one part of how a component can contribute or cause the Top Undesired Event. If a switch failure (closed in this example) can kill us all, so can inadvertent commanding of the switch closed. In this instance, the switch being commanded closed might be caused by some other failure (which we would miss if we develop the fault tree in terms of the switch failure). A person manually flipping the switch closed will have the same result! In this instance, flipping the switch closed might be called out in the procedure as a normal part of the operational sequencing.

So…What the Heck *IS* a Cutset? AND, What Good Is It?

Let's talk about Cutsets. Every paper and explanation I've read, every seminar and training class I've attended has done a poor job in explaining the Cutset Analysis and Cutsets. In every class, everyone's eyes glaze over who has never used cutsets in analyzing their trees. I will attempt to provide a good explanation.

Before you decide to skip this section be advised: If you use a cutset analysis on a fault tree, you are more likely to find something in the design which can cause a problem — something no one else has discovered. In other words, the System Safety

Engineer can finally show the other engineering groups that the safety engineer is worth something after all – that we can do more than push paper, kick tires and write papers about fault trees (or endless derivations of hazards analysis techniques)! The safety engineer has often been likened to those people in the Middle Ages who bayoneted the wounded after a battle was over (sort of like management and accountants). Cutsets give you an opportunity to be at the forefront of the battle—to actually have input into the design.

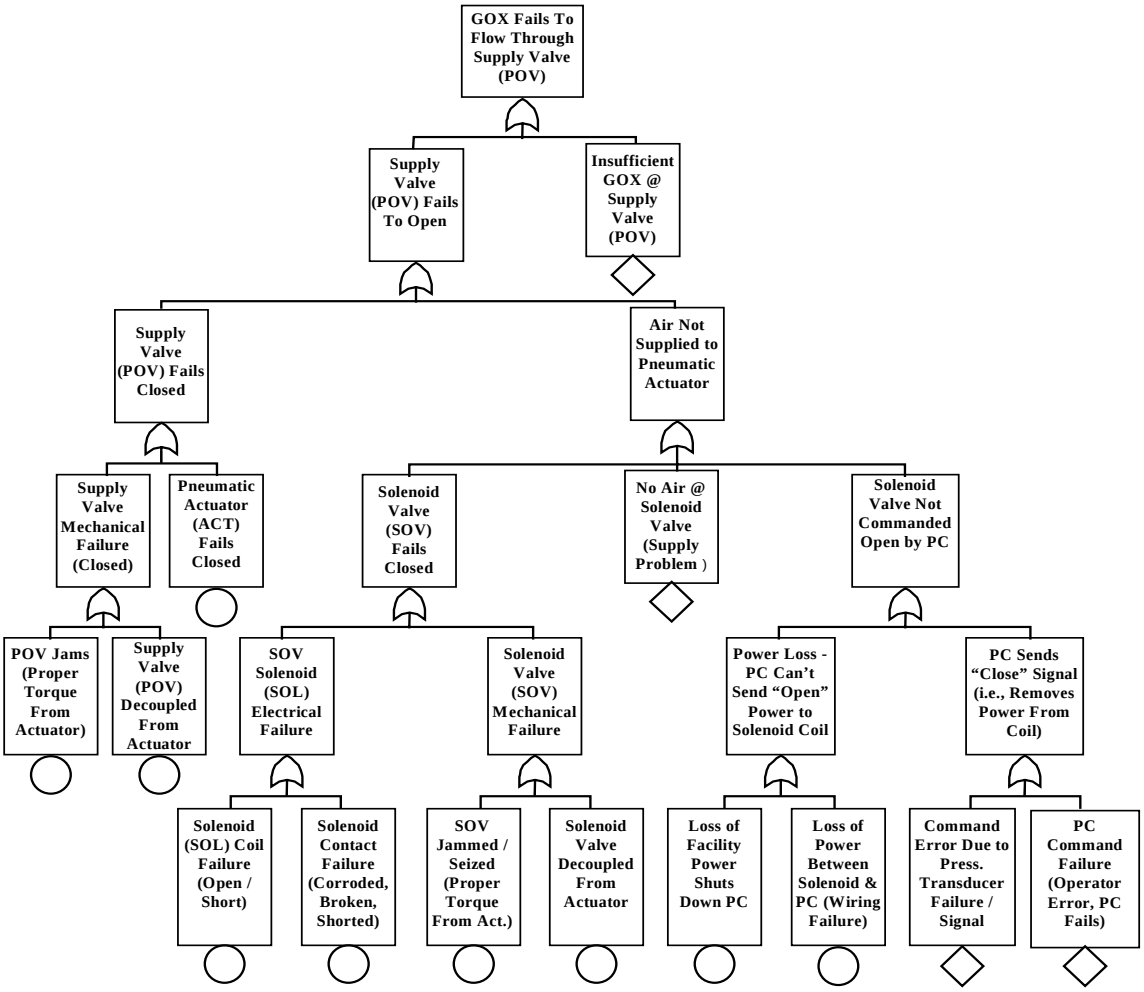

Figure 2

Are you excited yet? I thought so. Let's go!

The easiest way to think of a cutset is that each cutset is one scenario or set of events which causes the bad thing at the top of the tree to occur. Every tree has several cutsets, usually many cutsets, often hundreds or thousands of cutsets. A cutset can also be a single-point failure or event. Examples of cutsets:

If pushing the button causes us all to die, then "Pushing the Button" is one cutset in the fault tree.

If you have to remove the duct tape from a circuit and jiggle the blue wire to kill us all, we have a single cutset involving two events: 1) Removing the Duct Tape, and 2) Jiggling the Blue Wire." Let's call this cutset number two.

Different cutsets can also include different combinations of the same events. For example: if your tree shows that jiggling the blue wire and turning the red knob will kill us all, we now have a third distinct cutset, even though two of the cutsets include "jiggling the blue wire."

This may all sound very obvious. Well, it might be – but only in a small system with a small fault

tree. If you are analyzing a small system for which the fault tree only takes up two or three pages, you have either not developed the fault tree correctly, or you shouldn't be wasting your time on this methodology. (The obvious exception to this rule though is if management insists on a dog-and-pony chart). In a large tree for a complex system, these events may have been buried deep into the system. The duct tape, wire, and red knob may all be in branches located several pages from each other! You probably would not discover these combinations without performing the cutset analysis.

OK, what does finding these cutsets do for you?

It is the cutset analysis that can reveal the "Eureka's!" in a system. If the logic and naming convention are correct for the events, seemingly unrelated items often show up together. Even more striking, is that a system previously thought of as completely redundant or independent, can reveal single point failures which were not evident during initial design or other analyses. This is the main reason cutsets do not have to be quantified to have value.

Let's go back to our previous cutset example and look at the blue wire. Say, jiggling the blue wire shows up in 15 different cutsets. By controlling the jiggling of the blue wire, you have made it much less likely for any scenario involving the blue wire to occur. If you bolt down the wire, then all the scenarios would have to include: 1) "Bolts Have to be Removed," 2) "Jiggle the Blue Wire," and 3) "Remove Duct Tape," (or "Turn Red Knob…"). You have turned all 15 two-point cutsets into 15 three-point cutsets; i.e., it takes three things to occur instead of two for us all to die!

In a tree large enough to have the blue wire show up 15 times, the fact would be lost in the branches without performing the cutset analysis. Once a cutset analysis is performed, you can sort the cutsets to show any recurring events. Since the events in these cutsets may be buried deep within the tree, you might very well be the only one to find these events are even related! You may have just saved the day.

Another example could include our doomsday button. Before you laugh this off as a possibility, remember the Vacu-Lift venting system mentioned above. Usually, inadvertently cross-tying two subsystems is not as dramatic or simple as tying them directly to the same button (although it can happen). A subtle design flaw or a sneak that bypasses a safeguard (such as the "Arm" button) is more often the case. Such an occurrence happened on the Saturn program and was discovered as part of a Sneak Circuit Analysis. In this case a sneak bypassed the "Arm" switch and would allow the rocket to be fired by simply pressing the "Fire" button. A classical fault tree could also be used to discover such a flaw, if the tree is properly scoped and the analyst uses a nomenclature that is consistent throughout the fault tree.

Gate and Event Names Under Any Old Name Will Not Smell As Sweet

This brings me to my last point. *Consistent Nomenclature is CRITICAL.* An inconsistent or too rigid numbering system for gates and events can mean the difference between a tree that is sweet or a tree that stinks. Before we go on, let me explain what I mean by nomenclature. There are two components to a fault tree when using a good piece of FTA software: 1) description of what the failure is; and 2) the gate or event name (or label if you prefer). The name or label is what the software uses to perform the Boolean Algebra in calculating the cutsets. Labeling of the gates is not nearly as critical as is naming of events. However, consistency in the gates will allow you to replicate logic for similar but different components or subsystems. Consistent and logical nomenclature will also allow you to compare cutsets for symmetry. This is important when analyzing complex redundant systems. Looking for symmetry help you discover where you accidentally used an "AND" gate in place of an "OR" gate and vice versa.

Nomenclature that forces you to label the tree based on the tree structure will rarely allow you to find events that occur in multiple branches of the tree. A seasoned analyst *might* be able to structure the tree to circumvent this problem on a simple system. An example of this type of labeling would be the Work Breakdown Structure (WBS), i.e., 1, 2, 1.1, 1.2, 2.1, 2.2, etc. Use of a WBS or similar nomenclature will almost always force you into breaking the tree into engineering disciplines or areas of responsibility. This can be useful in assigning and tracking action items for an accident investigation, but will rarely provide adequate insight into a complex system.

If your naming convention is not consistent, single contributors to multiple branches or failures in a tree will be lost. For simplicity, let's say "Switch ABC fails Open (off)" contributes to 7 different failure scenarios in the tree. However, these 7 branches were developed on different days. Day one you give the failure a name of "Switch-abc-cl." The next day you give the same switch on a different part of the tree the name "SW-ABC-Closed." The next day it's "SW-ABC-CL" on branch 3 and so on. When you run the cutset analysis, the same switch would actually appear as 7 different switches (or however many different names you gave to "Switch ABC Fails Closed/Off"). At a minimum, this same switch contributes to no fewer than 7 scenarios. One inhibit or control (maybe even eliminating the switch) might help in 7 different failure scenarios. But, since you didn't label the event consistently, it looks like you have 7 different switches contributing to 7 different failures. In such a case, you might fail to suggest improvements/controls for the switch because it doesn't look like a major player. "If we have to spend money fixing 7 different areas, we will accept the risk" -- something I'm sure you've never heard from management.

Another common problem, which often causes confusion, is related to nomenclature, but is more often due to ignorance as to how the fault tree works. In replicating logic from one part of a tree to another, untrained analysts and neophytes often use the same nomenclature for similar but different components. For instance, say we have two switches ABC, and DEF, which are exactly the same *type* of switch but are *physically different* switches. Instead of using "SW-ABC-Fails- Open" and "SW-DEF-Fails-Open," the analyst uses "SW-Fails-Open" in both places. A cutset analysis on this fault tree would show the same failure contributing to several scenarios, when in fact the failures are caused by different components. This can be an egregious error if your system were to have a bunch of different switches.

Conclusion

Performing FTA is a craft that requires the proper tools and knowledge of logic structure and systems design. Although there are many ways to use FTA, in the space community few trees are developed that fully utilize the power of this excellent tool. To fully realize the FTA potential, the analyst must: 1) properly (and narrowly) define the Top Undesired Event, 2) arrange the tree into scenarios rather than "failures," 3) use a consistent nomenclature to prevent confusing multiple failures as one failure or vice versa, and 4) use a computer program to perform cutset analysis. Above all, the analyst must be able and willing to work with a wide variety of engineering disciplines and subsystems. This includes the ability to see how the pieces fit in a system and to properly analyze the interactions at the interfaces.

Biography

Allen Long, Senior System Safety Engineer, Hernandez Engineering, Inc., MSFC, AL 35812 USA, Telephone – (256) 961-1177, facsimile – (256) 544-8022, e-mail – allen.long@fault-tree.net

Allen Long is a Senior System Safety Engineer for Hernandez Engineering, Inc. on the Safety and Mission Assurance Contract at Marshall Space Flight Center (MSFC). He specializes in hazards analysis FTA for systems and process design. Mr. Long is considered the MSFC resource person for FTA and regularly performs FTA for Development Programs as well as for existing systems and mishap investigations. Past projects have included X-33 Linear Aerospike Engine and Ground Support Systems design, Chandra X-Ray Telescope, Gravity Probe B, Transfer Orbit Stage, Solar Thermal Flight Experiment (Shooting Star), and the International Space Welding Experiment. He has been responsible for FTA on the majority of mishap investigations at MSFC for the past ten years on programs including both Tethered Satellite System (TSS) missions, Alternate Turbopump, External Tank, Shuttle Main Engine, and Test Stand Operations associated with several programs. Prior to working at MSFC Allen worked for the Hazards Analysis Department at Thiokol in Brigham City Utah. While at Thiokol, Allen co-developed Thiokol's Handbook for Performing Hazards Analysis, and performed numerous hazard analyses and FTA. He was also responsible for the fault tree effort for the Peacekeeper Core Removal Fire Investigation at Thiokol.