

Variants of Classical Cutset Characterization

R. Allen Long; Rockwell Collins Government Systems; Cedar Rapids Iowa

Keywords: Minimal Cutsets, Fault Trees, Boolean, Nomenclature

Abstract

Many attempts have been made to explain the concept of cutsets and cutset analysis to fault tree neophytes and the managers who make decisions based on fault tree analysts' analysis. Those new to the discipline have often been overheard lamenting, "Cutsets (?) . . . Is this instructor speaking in tongues?" Keeping in mind these timeless questions, this paper (a follow-up to the author's *Beauty & the Beast – Use and Abuse Of The Fault Tree As A Tool* – Ref. 1) is another futile attempt to provide the reader a clear understanding of Super Cutsets, Minimal Cutsets, and Cutset analysis. Cutset analysis can be instrumental in finding unexpected problems in complex systems and is just as useful for qualitative fault trees as it is for quantitative fault trees. The relationship between cutset analysis and naming conventions (both proper *and* improper) is explained. The author discusses two common mistakes and how to prevent them by using proper nomenclature and cutsets. Although this paper is written primarily for fault tree newcomers (and the author's own entertainment), the information should prove useful to even the most seasoned of fault tree analysts. You never know when someone will ask you, "What the heck is a cutset?"

Introduction

Ever since Fault Tree Analysis (FTA) became an international phenomenon, fault tree neophytes, managers who oversee fault tree practitioners, and the extreme few members of the general populace who even care have asked, "What is a Cutset?" and, "Why do I care?" Having been a neophyte (on more than one occasion myself), I have been subjected to cutset explanations which, being the "soft-spoken & un-opinionated" man that I am, I can only describe as indescribable. At the risk of being thrown out of the fault tree club for exposing the man behind the curtain, I will attempt to reveal the elusive *understandable*, explanation of the inscrutable Cutset.

The Cutset Explanification Complication Principle (Science Meets Snake-Oil)

Fault Tree Practitioners have devised egregiously complex explanations in order to appear more technically adept than perhaps we really are. Cutset explanations have been distilled into a kind of white magic incantation if you will, and peddled by the sorcerers of systems – the modern equivalent of witch-doctors (but not as well-dressed). Don't confuse this with the black art of reliability prediction and probabilistic risk assessment. This leads us to one of Long's Laws:

"Never use a 25-cent word when there is a perfectly acceptable \$4.00 word available. If you can't find a suitably confusing word, make one up."

(How do you think I became a gang member in such an elite club in the first place?)

Cutset Instruction Deconstruction

The diabolical thing about cutsets is the simplicity of cutset *definitions*. Explaining how cutsets work and why they are important, however, is rarely done well (and this paper is no exception). Instructors can even take a bit of sadistic pleasure from traumatizing the fault tree trainee (I know I do). Sadly, I've seen fault tree analysis taught by those who (sort of) know the theory but have never performed fault tree analysis on a real-world application – let alone, one of any complexity.

One fun way to torture the uninitiated is to teach them how to calculate cutsets by hand *before* or (better yet) *without* teaching them how to properly model a system using fault tree logic and the power of cutsets. Jump straight to the math before tree-o-phytes can even begin to comprehend cutset basics. Forget telling tree toddlers that fault trees small enough for hand calculation are too small for any meaningful system analysis in the real world. This teaching method can have unexpected benefits. An ex-manager once told a coworker and me that we didn't need fault tree software for fault tree analysis on a new contract, because HE had calculated cutsets by hand in class (he didn't say whether he used his fingers).

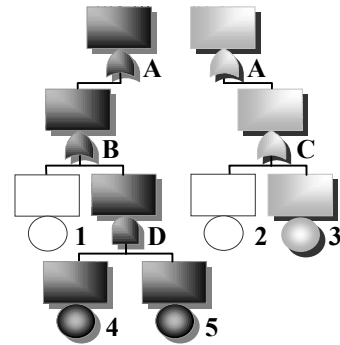


Figure 1 – The Path to Cutset Destruction Instruction

Who has not been terrorized by an introductory “cutset” figure like the one in Figure 1? Legion are those who now think a cutset is a path to the tree top. “*A Cutset is NOT a Path.*” Even without meaning to, instructors who use paths to illustrate cutsets early in fault tree instruction can confuse students into thinking cutsets and cutset paths are the same thing. I was introduced to cutsets this way and came away without a clue as to what a cutset was. And now, I am writing about them (SCARY!!) . . . Frightened yet? You should be!

These methods are a bit like teaching beginners a foreign language by starting with formal tenses and word forms, before teaching basic language survival skills. *But wait, there's more!* For a limited time you can read on for a better way (Well . . . OK . . . a *different* way) to learn cutsets!

Cutset definitions range from “Simple & Sublime” to “Complex & Obscure”. Ironically, the more complex the definition, the less likely anyone who understands it actually needs a definition of cutsets in the first place. Here are some examples of classical cutset definitions:

- *A cut between source node & sink node such that any proper subset of such Minimal Cutset is not a cut.*
- *Accident sequence failure combinations. (Perhaps, the most simple definition ever created)*
- *A set of components such that, if all components in the cutset fail, the undesired TE occurs.*

The Top Event (TE) and Top Undesired Event (TUE) are fancy titles for, “*The Really, Really Bad Thing At The Top Of The Fault Tree*”. (If it weren't really, really bad, your company wouldn't give you bags of money to perform fault tree analysis to begin with).

Let's Cutset to the Chase

The easiest way to think of a cutset is that it is one set of events that make the bad thing at the top of the tree happen

Technically, when we refer to “Events” we are talking about “Basic Events” – those events at the bottom-most level of a fault tree. Gates are sometimes referred to as Intermediate Events. For purposes of this paper, “Events” and “Basic Events” are interchangeable. Gates will be Gates.

- An Event is anything, any cause, that helps to make the Top Event to come true!
 - **Events are not limited to failures.**
 - **Faults, design errors, mistakes and, yes, failures can all be events.**
 - **Normal operations can sometimes be events that contribute to the Top Event.**
- A cutset can be a single-point failure or a single event or fault.
 - Confusing isn't it? I don't know about you, but my wife and, therefore, I would be upset if her new dining room set was delivered with only one piece! So . . . think of it this way:

**If any set (including a cutset) shows up with only one item in it,
you need to do something about it.**

- The same component may show up in more than one cutset (often in dozens of cutsets).
- Every useful fault tree has more than one cutset – usually hundreds or thousands of cutsets.

But of course, I am about to show you a useless fault tree with only one cutset.

Domesticating the Untamed Cutset

Let’s say, for instance, we have a prankish paint-can firing circuit with an “ARM” switch and a “FIRE” switch. Both switches must be “ON” at the same time to dump the paint bucket on our unwitting practical joke participant. If both switches are inadvertently activated, the wrong person (that would be you) could be instantly turned into a clown. Figure 2 shows the fault tree constructed using our detailed system description: (In the mean time, please stand over . . . *here*).

For our example we have exactly **One Cutset** that consists of two basic events:

“ARM ON” & “FIRE ON” (formally known as “ARM ON” • “Fire On”)

This Cutset shows that it takes two events to cause the top event and is called a two-order cutset. (**One event = One-Order Cutset, Two events = Two-Order, Three events = Three-Order . . .**)

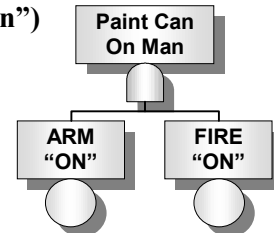


Figure 2 – One Pitiful Fault Tree Example

In the spirit of fault tree methodology, I will now lead you from the general (read, “vague generalities”) to the specific (i.e., Why did I not tell you this in the first place?)

Quantity-TATIVE versus Quality-TATIVE – Why the Numbers Don’t Always Add Up

Although Cutsets are important in *quantitative* fault trees (i.e., trees that compute probabilities), *quantitative* fault trees without cutset analysis can still be useful. Some fault tree software packages can calculate and show probabilities at the basic event, gate, and top event levels. I know several fault tree practitioners who use gate probabilities exclusively in lieu of cutset analysis – typically for validating whether a design meets certain predetermined requirements. However, in using this methodology, you might find yourself explaining to a customer or manager why the gate probabilities under an “AND” gate do not add up to the Top Event Probability. More on this later.

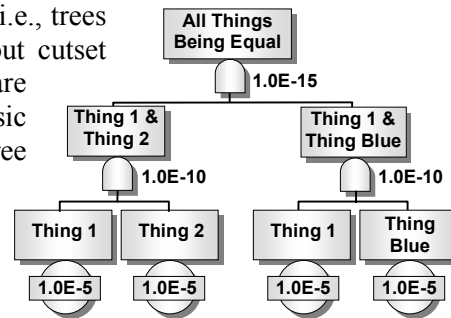


Figure 3 – Fault Tree with Seussian Syntax, Geisel Gates & Improbabilities

Figure 3 shows $1.0 E-5 \bullet 1.0E-5 = 1.0E-10$. So far so good. Therefore, $1.0E-10 \bullet 1.0E-10 = 1.0E-15$? Hey! . . . What’s going on here?

You just witnessed a minimal cutset reduction production! I was recently asked about this very conundrum (which is why we are talking about it now). Without having seen the fault tree, it was obvious there was something common to both branches under the “AND” gate. Quantified fault trees without cutset analysis do not exploit the full potential of the methodology (if only because the act of evaluating cutsets helps you understand how cutsets work and why).

When developing *qualitative* fault trees, cutset analysis is the **ONLY** way to properly evaluate a complex system. Without cutsets, a qualitative fault tree is no more than a Pseudo-Scientific-looking “Dog-and-Pony” chart. **Fault trees do not have to be quantitative to be effective.** Qualitative fault trees are extremely useful for discovering design flaws – but only if proper

cutset analysis is performed. Those of you performing fault trees on systems without formally calculating cutsets are probably already determining cutsets in your head. You just don't realize it. A form of informal formal method, so to speak! Your method is madness because you are undoubtedly missing some of the most important cutsets.

The Capable Cutset – Cornerstone of Fault Tree Analysis

Enough of the generalities. What does that all mean? And . . . how dare I sneak in another confusing term, **“MINIMAL Cutset”**? Generally, when someone refers to *Cutsets* or *Cutset Analysis*, they are really talking about *Minimal Cutsets*. I will attempt to explain the moxie of Minimal Cutsets a bit later. For now, here are the “Cutset CliffsNotes™”:

Cutsets can reveal:

- Single-point failures, faults or events in systems where you thought you had redundancy
- Areas needing new or additional controls
- Failure to implement controls you thought were in the design
- Components that repeatedly show up as contributing to the top event
- System design flaws that allow properly functioning and/or correctly operated equipment
- Whether you constructed the fault tree correctly

To better explain these benefits, let's develop a fault tree for the system in Figure 4.

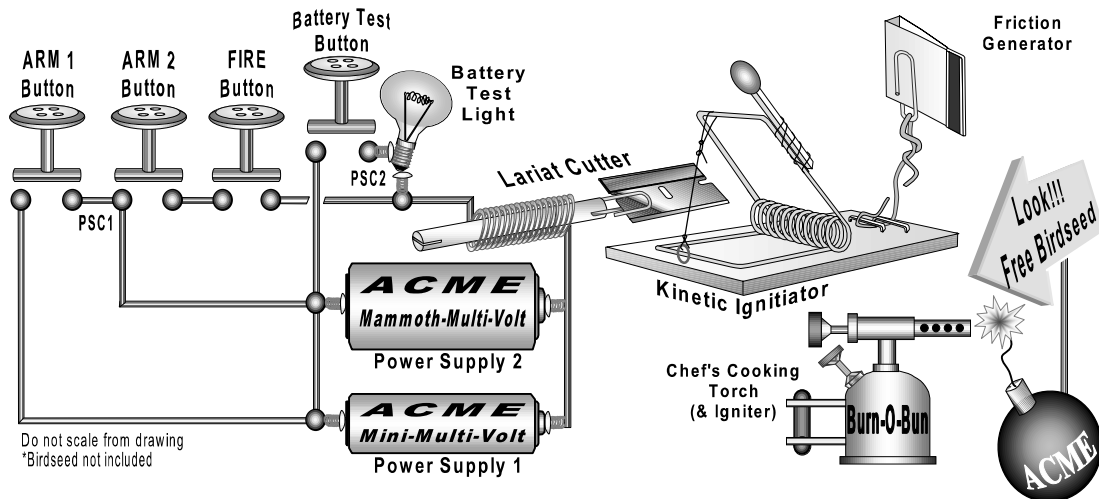


Figure 4 – ACME Arid Avian Extinctifier*

Figuratively Speaking, “How Does this Contraption Work?”

The devilish device in Figure 4 (confiscated off a coyote in the Arizona desert) is used for controlling the desert bird population. Officially known as the ACME Arid Avian Extinctifier (A³E), it is often called the Desert Bird Catcher. We are fortunate to have the general areas of two Pesky Sneak Circuits (PSC1 & PSC2) noted. Normally we are not so lucky and must use Fault Tree Analysis to find these hidden gems. The system description and *intended* operation are provided below. By using the actual design in Figure 4, the fault tree in Figure 5 will reveal this good intention as paving the way to the place where snowballs have no chance.

- **“ARM 1”**, **“ARM 2”**, & **“FIRE”** Buttons must all be pressed (*in theory*) to fire up the A³E.
- Power is supplied to the Lariat Cutter by redundant batteries (Power Supplies 1 & 2).
- When the Lariat Cutter solenoid is activated, the Lariat is cut, which releases the mousetrap spring, which strikes the match against the matchbook.

- The lighted match ignites the igniter setting off an explosive device of your choosing.
- A Battery Test Button and light are included.

Fault Tree Fun-damentals

Figure 5 shows the fault tree for the Desert Bird Catcher in Figure 4. *Every* fault tree should include “Groundrules & Assumptions”. Even for our small example, we have a few groundrules:

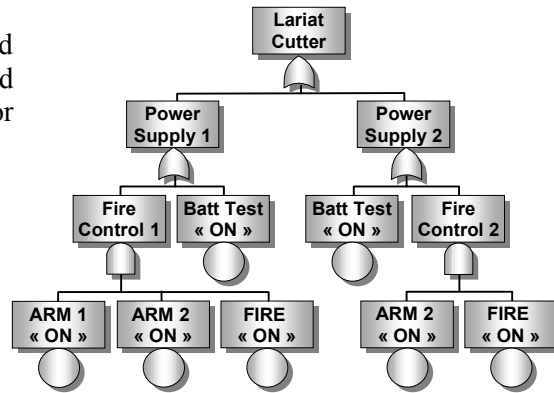


Figure 5 – AAAEA Fault Tree

1. The Burn-O-Bun igniter can only be lighted by the Match from the Ignitiator. (Spontaneous ignition and ignition from outside sources are not addressed).
2. We are only interested in accidental power to the Lariat Cutter. Operators on the shop floor immediately spotted three single-point Lariat-related failures. ACMEngineering wrote a policy forbidding any single-point failure from occurring (as opposed to correcting them). Operators fixed them anyway.
3. Batteries are assumed to have power. Scenarios involving dead batteries are not modeled.

There are only two minimal cutsets in the A³E fault tree and they look like this:

Cutset 1 = “**Battery Test On**” and Cutset 2 = “**ARM 2 ON**” • “**FIRE ON**”

Why are there are so few cutsets in both trees? Where is the cutset with three events? After all, there is an “AND” gate with three events under it. The answer my friend, is not blowing in the wind. It is Minimal Cutset magic in the end.

They’re Delightful – They’re de-Magic – They’re de-Minimal Cutsets

I have yet to find or invent a really good explanation of “minimal” cutsets. Being the optimist that I am, (in a skeptical, cynical sort of way) I will dare to attempt this dangerous feat, but only with the aid of *exemplifications*. Using only smoke-&-mirrors, I shall clear away the shroud that . . . well . . . *enshronds* the mysterious Minimal Cutset! For starters:

A “Non-Minimal Cutset” is referred to as a “Super Cutset”, which is not as super as it may sound.

A Minimal Cutset is determined by one or more of the following operations:

Events appearing to be different but are actually the same are combined and counted as one.

If a one-order cutset shows up in multiple areas of the fault tree, it only shows up as one cutset.

In Figure 4, “**Battery Test ON**” shows up twice – once under “Power Supply 1” and again under “Power Supply 2”. This is the same event – not two separate events. Pushing the Battery Test Button only once provides power through both power supplies at the same time.

Duplicate cutsets (all with the same events) are purged (only one of these cutsets is counted).

Duplicates are really the *same cutset showing up in different parts of the tree*. Therefore, the cutset is counted only once (and for good reason).

Let's say "ARM-1-ON" • "FIRE-ON" shows up in five different areas of the fault tree. The cutset analysis would show only one set of these two buttons. You don't have to press both buttons five times to fire up all five subsystems. Wouldn't you rather be alerted that pressing these buttons only once will fire all systems at the same time?

A quick word about the confusing "Cutset versus Cutset Path" part: Although there is only one minimal cutset, it can show up in several places on the tree. Each location will have a separate path through subsystems and scenarios to reach the top event. Following each path is a good way to determine effectiveness of existing controls and why expected redundancies may have failed.

We also want to find all of the cutsets which contain the minimum number of events needed to cause the Top Event. This is not to be confused with multiple-point failures.

A cutset with multiple events is eliminated if one of its events shows up in a single-order cutset (a cutset with only one event or failure). The same is true, for instance, if a third-order cutset (a cutset with three events) has two of its three events show up in a two-order cutset (a cutset with two events). In this case, the cutset with the three failures is eliminated.

For example, in Figure 5, one Super Cutset (A.K.A. a non-Minimal Cutset) is: "ARM 1 ON" • "ARM 2 ON" • "FIRE ON" (all three are under the "AND" Gate under Power Supply 1).

1. Activation via Power Supply 2, however, only takes "ARM 2 ON" • "FIRE ON".
2. So . . . "ARM 1 ON" needs "ARM 2 ON" & "FIRE ON" for it to help activate the Lariat Cutter (it still needs help from "FIRE ON" & "ARM 2 ON" to do the deadly deed).
3. Since "FIRE ON" knows it doesn't need two "ARMS" to cause mischief, it decides to throw poor "ARM 1 ON", out of the cutset club. All that is left is "ARM 2 ON" • "FIRE ON".
4. Bet you don't think this cutset is so super now!

If a single event shows up twice in the same Super Cutset, the Minimal Cutset will be a single-point failure or, a cutset of one.

Let's say the system actually requires power from both batteries to activate the Lariat Cutter.

First off, you should be ashamed for not catching this design feature in the first place. Secondly, in Figure 5a, we have substituted an "AND" gate in place of the "OR" Gate under the Top Event to correct your modeling error.

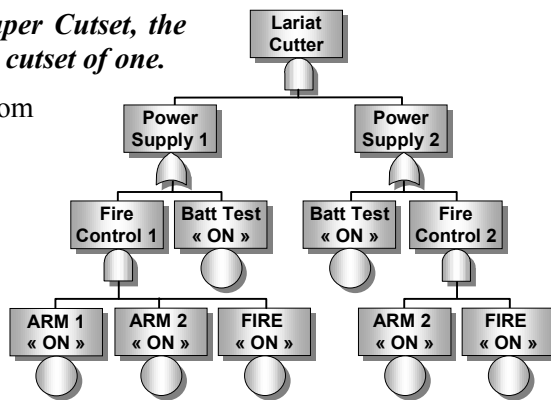


Figure 5a – Modified Fault Tree (With the "AND" Gate at the top)

Now we have two Minimal Cutsets:

1. "ARM 1 ON" • "ARM 2 ON" • "FIRE ON"
2. "Battery Test ON"

The Super Cutsets look like this:

1. "ARM 1 ON" • "ARM 2 ON" • "FIRE ON" • "ARM 2 ON" • "FIRE ON"
2. "Battery Test ON" • "Battery Test ON"

Looking at the first super cutset, we notice two duplications – "ARM 2 ON" and "FIRE ON". Both show up twice in the same super cutset and are thrown out as superfluous.

In Super Cutset Number 2, "Battery Test ON" shows up twice (i.e., "Battery Test ON" • "Battery Test ON"). Even though these show up under different sides of the tree, they are one

and the same. Minimization reveals that “**Battery Test ON**” is a *single-point failure (or event)*. Without minimization this single event could have easily be mistaken for a two-point failure set.

When properly constructed, a fault tree can discover problem areas other methods may miss (and often do).

This last example demonstrates just how important Minimal Cutsets are to the *analysis* part of Fault Tree *Analysis*. In complex systems, even those designed to be redundant, single-point failures can be hidden throughout the system. Control systems are especially prone to having single point failures since redundant systems usually use a common control system. Minimal Cutsets can help identify areas where supposedly isolated or separate systems are actually tied together. Rarely do these common pieces identify themselves until they are coaxed out using cutset analysis. Modeling complex real-world systems usually involves wading through and tying together numerous system drawings and schematics to understand the system.

Mambo Madness With Multiple-Event Cutsets! (Worth The Price Of Admission)

Just about everyone (from the giggliest school-girl – who really couldn’t care less about all this, to the most bothersome bean-counter) knows single-point failures are bad. **However, a shortage of single-order cutsets does not mean the system is Ready-to-Rumba** (if I may mix my dance metaphors). You still might have a fallacious fault tree due to errant event names or gross gate gaffes (not to be confused with the guffaws of your peers). More on naming problems later. If there truly are no single-order cutsets, the remaining cutsets may still tell a tale – if you know how to read the tea leaves. A component showing up in a boatload* of cutsets is warning you there may a common design problem in the system (*boatload; a technical term). Common components in multiple *two-order* cutsets deserve special attention.

Suppose you have a fault tree with 21 two-order cutsets – each having an event common to all 21 cutsets. If this common event fails, only the single remaining event in any one of the 21 cutsets has to occur to cause the top event. Though these are not one-order cutsets, such a common event *will* increase the overall probability of occurrence – perhaps by orders-of-magnitude. In some cases, two-order common-event cutsets may even overshadow the probability and risk of single-point failures in the system. In other words, the common event is begging for your attention.

Conversely, a common event may be a control you *want* to show up in a *beaucoup** of multiple-event cutsets (*beaucoup; French technical term for boatload). Cutsets sharing a common event should require at least *three* failures. In this case looking at the two-order cutsets can point out:

- Places in the fault tree where you accidentally omitted a control (i.e., there are two-point failures in the cutsets when you know there are really three-point failures in the design).
- Areas in the design that did not properly implement a control.
- Additional design areas where controls could turn two-point failures into three-point failures!

The importance of Minimal Cutsets should not be minimized, even though the cutsets themselves must be minimized.

Cutset Follies and Fault Tree Foibles
(Boolean Bloopers and the Farcical Fault Tree)

The most common mistakes in FTA are due to misunderstanding and mishandling basic events.

- Similar, redundant, (but physically distinct) components must be referred to differently to prevent multiple-point failures from looking like a single-point failure.

- A given event must be referred to the exact same way throughout a fault tree (when used several times) to avoid turning a single event into several different events on the fault tree.

This mistake makes a single-point failure look like a multiple-point failure. As such, an unsafe system could be released for production and use – perhaps with catastrophic consequences.

Better Living Through Nomenclature

Two things will help you keep components and events in their proper place on the tree:

- Understanding and using consistent nomenclature – *Properly Naming Events and Gates throughout the fault tree.*
- Understanding how nomenclature fits in with cutset analysis – *Knowing how to properly Name Events to ensure accurate cutset analysis results.*

Unfortunately, these two concepts are often not explicitly discussed in fault tree courses, “How To” guides, or references. Rather, beginners are left to infer these concepts through the mathematics of calculating cutsets by hand. You don’t have to know the intricacies of probability theory and the more arcane aspects of Boolean algebra to properly model a complex system. Personally, I can’t balance a checkbook (at least that’s what I tell my wife).

The Nature of “Nomenclature” versus “Description” of Details

Fault Tree Analysis (FTA) Software typically uses two labels:

“**Description**” explains what is happening at the gate or event (i.e., the text inside the big box).

“**Name**” is the juju that software uses to work its mojo.

- The “Name” is the Boolean label used by the software perform cutset analysis.
- This is also the portion of the gate/event the software uses for cut & paste, logic replication, and gate transfers. Figure 6 shows three examples of how several popular fault tree programs display the **Description** and **Name**.

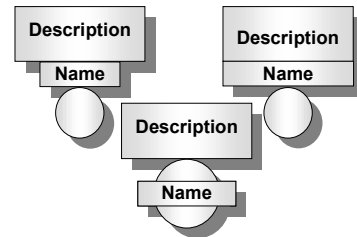


Figure 6 – FTA Software: Name versus *Description*

Nomenclature originates from the Geek words:

- “**Noman**” as in “No man on the football team wants to touch my slide rule” (usage has since evolved to “Pocket Calculator” and again to “Pocket PC”).
- “**Claptrap**” defined as “Insincere or Pretentious Language”.

In other words, “Nomenclature” is the practice of inventing naming schemes that are both scary sounding and incomprehensible to all but the most technically obsessed and socially unaware. Fortunately, a few of us are technically adept enough to *pose* as techno-cats and technocrats and still understand the concept of personal hygiene, social interaction outside of the chat-room and what the “*little fork*” is used for. Thus, we are occasionally able to translate the arcane into “Ah Ha”. We are the “Somewhat Highly Comprehended Technical Precocious” (*Hey . . . You . . . Over there . . . Pay no attention to that flashing VCR*).

Simply stated: “**Nomenclature is the act of Naming Names**”

“Different” is Good – Unless It’s the Same

I touched briefly on two of the most common fault tree errors – both caused by inconsistent event names. I also spoke about Boolean logic, but only in passing. Here is where the two concepts come together. Although Boolean logic takes into account how the events fall under the gates in

order to calculate cutsets, the logic ultimately depends on Event Names to keep things straight. **Event Names are just as important as proper Gate Logic for correctly calculating the cutsets.** Boolean logic cannot tell if you called a single event two different names, or whether you called two events same name – any more than it can tell if you used the wrong gate type.

I know I stressed earlier that events are not always components. However, the majority of fault trees benefiting from cutset analysis normally end up dealing with components at some level. Even when modeling human interaction, the end results are usually based on the act of *commanding components to activate or deactivate (or failure to do so when required)* (Figure 7). This typically involves button-pushing, lever pulling, switch switching . . . You get the idea. I mention all this to rationalize my emphasis on failures of components in the examples below. However, the rules discussed in the examples are valid and useful for all event types.

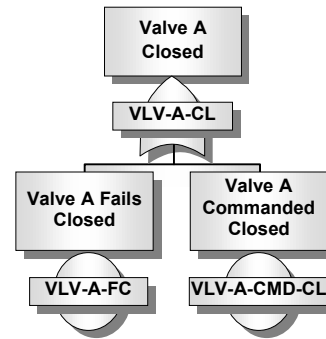


Figure 7 – Command versus Failure

Different Events Must Have Different Event Names

Figure 8 illustrates the common mistake of using identical names for different components. In this example, the analyst intended to show that each of 12 different and physically distinct valves can fail in the same way. If properly modeled there would be 24 cutsets – one solenoid and one valve failure for each of 12 valves (A thru L). As it is modeled now, there would be only two cutsets with single-point failures (i.e., two single-order cutsets):

Cutset 1 = "VLV-FC" & Cutset 2 = "SOL-FAIL"

A Component Must Keep the Exact Same Name Throughout the Entire Fault Tree

The naming conventions in both branches in Figure 9 are correct – but only if they show up in different fault trees altogether and not together in the same fault tree. Are we all together on this?

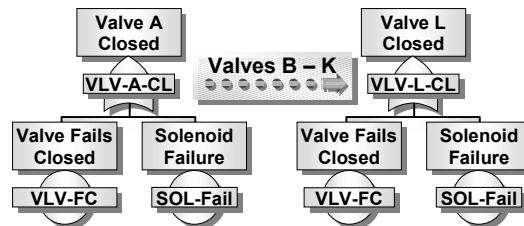


Figure 8 – Same Name + Different Components = Bad Method In Your Mojo

If you are lucky, both of these branches will fall only under "OR" Gates, thus creating four one-order cutsets (i.e., single-point failures):

1. "SOV-A-FC" 3. "SOL-A-FC"
2. "A-SOV-FC" 4. "A-SOL-FC"

This *Fault Pas* is likely more humiliating than dangerous. Multiplying a single-point failure by "How-Ever-Many-Different-Names-You-Called-Something" will at least get your attention. You never know, with the right project and publicity, your management might even learn of your results from a television news crew!

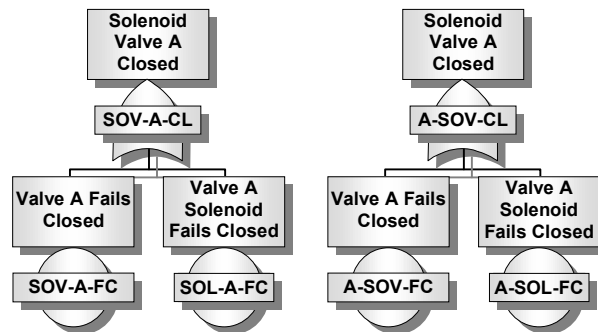


Figure 9 – Same Component + Different Names = Potential Catastrophic Fault Tree Error

Hyperbole aside, if the same component shows up with several different names under a combination of "AND" gates, cutset analysis would show this event as multiple failures – *not the*

single failure it really is! In a large tree, finding this problem will not be easy. This serious mistake could lead to operation of a system based on the belief there are no single point failures. In all fairness, this mistake is common, even for those of us who know better (if you saw how I type, you'd be amazed that I ever get a fault tree to come out right). As is the case for most things in life, the answer to this problem is "*Balance and Symmetry*".

Finding Symmetry and Serenity in a Cutset World

Comparing cutsets can help you find mistakes such as using the wrong gate type, or using two different names for the same event. We are looking for two equally important anomalies – *Symmetry where there should be none*, and *Lack of Symmetry where symmetry was expected*.

If your system is clearly redundant and you have a one-order cutset such as "*Side A Valve Fail Closed*" you probably used an "OR" gate where an "AND" gate belongs. In this case you expected cutset symmetry: "*Side A Valve Fail Closed*" • "*Side B Valve Fail Closed*"

A two-order cutset such as "*A Side Valve Fails Closed*" • "*Side A Valve Fails Closed*" reveals different names were used for the same component. Renaming the events with the same name will yield a single-order cutset of "*Side A Valve Fails Closed*". On the other hand, this cutset might have been telling you to re-label the event to "*Side B Valve Fails Closed*". You would be surprised how a quick scan of two-order cutsets can often find a lack of symmetry *within the wording* of events in a two-order cutset – even in cases where a two-order cutset *is* expected.

Unfortunately, you'll have to wait for the third paper in this series for more insights on symmetry. Until then, ready yourselves for cutset paths. By then, I might even be able to talk about them!

FINALLY, THE FINE PRINT: No animals were harmed as a result of this paper (although somewhere in Arizona there is a hungry coyote). Nor, have any facts stood in the way of Allen's opinions.

References

1. R. A. Long, Beauty & the Beast – Use and Abuse Of The Fault Tree As A Tool, Proceedings of 17th International System Safety Conference, pp. 117-127, 1999.
2. C. A. Ericson II, Fault Tree Analysis By Design, Proceedings of 16th International System Safety Conference, pp. 121-126, 1998
3. J. D. Andrews, Fault Tree Analysis – Common Misconceptions, Proceedings of the 20th International System Safety Conference, pp. 401-409, 2002

Biography

R. A. Long., Rockwell Collins Government Systems, 400 Collins Road, Cedar Rapids, IA 52498
telephone - (319) 295-7687 facsimile - (319) 295-3775, e-mail allen.long@fault-tree.net.

Allen has worked in aerospace safety for over 20 years and has specialized in fault tree analysis for around 17 years. He has written several papers and was awarded "Best of Conference Paper" and "International System Safety Engineer Of The Year" in 1999. Most recently, Allen and co-author Bill Vesely published *The Role of Fault Trees in Probabilistic Risk Assessment* in the Journal of System Safety (4th Qtr 2002). Allen relives his past accomplishments through his fault tree website at <http://www.fault-tree.net>. **Allen resides in Cedar Rapids with his wife and children who all take him about as seriously as you should.**