Introduction To Fault Tree Analysis

Clifton A. Ericson II
Design Safety Solutions LLC
ciftonericson@cs.com
FTA Outline

1. FTA Introduction
2. FTA Process
3. FT Terms/Definitions
4. FT Construction
5. FT Construction Rules
6. FT of Design Models
7. FT Mathematics
8. FT Evaluation
9. FT Validation
10. FT Pitfalls
11. FT Auditing
12. FTA Application Examples
13. FTA of Example Systems
14. FT Codes
NASA has been a leader in most technologies it has employed in its programs over the years. One of the important NASA objectives is now to add Probabilistic Risk Assessment (PRA) to its repertoire of expertise in proven methods to reduce technological and programmatic risk.

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

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

- Evaluates complex systems (small to large)
- Identifies causal factors that can result in an Undesired Event
- Visual Model - displays complex cause-consequence combinations
- Combines failures, errors, normal events, time, HW, SW, HE
- Deductive (general to the specific)
- Provides risk assessment (Quantitative / Qualitative)
- Defined, structured and rigorous
- Easy to learn, perform and follow
- Utilizes Boolean Algebra, probability theory, reliability theory, logic
- Proven over time
Example FT

System

Battery [Diagram]

System Undesired Event: Light Fails Off

FT Model

[Diagram]

Cut Sets

Event combinations that can cause Top Undesired Event to occur

<table>
<thead>
<tr>
<th>CS</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$P_A=1.0 \times 10^{-6}$</td>
</tr>
<tr>
<td>B</td>
<td>$P_B=1.0 \times 10^{-7}$</td>
</tr>
<tr>
<td>C</td>
<td>$P_C=1.0 \times 10^{-7}$</td>
</tr>
<tr>
<td>D</td>
<td>$P_D=1.0 \times 10^{-6}$</td>
</tr>
<tr>
<td>E</td>
<td>$P_E=1.0 \times 10^{-9}$</td>
</tr>
</tbody>
</table>
**FT Building Blocks**

- **Primary Failure**
- **Secondary Failure**
- **Normal Event**

**Basic Events**
- Basic Events
- Gate Input
- Gate Output

**Gates**
- OR Gate
- AND Gate
- Inhibit Gate
- Exclusive OR Gate
- Priority AND Gate

**TUE**

**Condition**

**Transfer**

**Text Box**
FTA – Deductive Approach

Analyzing from the Undesired Event to the root cause(s).

Going from the general to the specific.

Only the components that contribute to UE.
Two Types of FTA

- Proactive FTA
  - FTA during system design development
  - Improve design by mitigating weak links in the design
  - Prevent undesired events and mishaps

- Reactive FTA
  - FTA during system operation
  - Find root causes of a mishap/accident
    - Modify the design to prevent future similar accidents
FTA Coverage

- Hardware
  - System level
  - Subsystem level
  - Component level
  - Environmental effects

- Software
  - System level control
  - Hardware/software interface

- Human Interaction
  - Human error
  - Human performance
  - Organizational structures

- Procedures
  - Operation, maintenance, assembly

- System Events
  - Failures Events
  - Normal Events
  - Environmental Events
**FT Strengths**

- Visual model -- cause/effect relationships
- Easy to learn, do and follow
- Models complex system relationships in an understandable manner
  - Follows paths across system boundaries
  - Combines hardware, software, environment and human interaction
  - Interface analysis - contractors, subsystems
- Probability model
- Scientifically sound
  - Boolean Algebra, Logic, Probability, Reliability
  - Physics, Chemistry and Engineering
- Commercial software is available
- FT’s can provide value despite incomplete information
- Proven Technique
Why Do A FTA?

- **Root Cause Analysis**
  - Identify all relevant events and conditions leading to Undesired Event
  - Determine parallel and sequential event combinations
  - Model diverse/complex event interrelationships involved

- **Risk Assessment**
  - Calculate the probability of an Undesired Event (level of risk)
  - Identify safety critical components/functions/phases
  - Measure effect of design changes

- **Design Safety Assessment**
  - Demonstrate compliance with requirements
  - Shows where safety requirements are needed
  - Identify and evaluate potential design defects/weak links
  - Determine Common Mode failures
Example FTA Applications

- Evaluate inadvertent arming and release of a weapon
- Calculate the probability of a nuclear power plant accident
- Evaluate an industrial robot going astray
- Calculate the probability of a nuclear power plant safety device being unavailable when needed
- Evaluate inadvertent deployment of jet engine thrust reverser
- Evaluate the accidental operation and crash of a railroad car
- Evaluate spacecraft failure
- Calculate the probability of a torpedo striking target vessel
- Evaluate a chemical process and determine where to monitor the process and establish safety controls
FTA Misconceptions

- FTA is a Hazard Analysis
  - Not true
  - Sort of meets definition of hazard analysis (HA), but not a true HA
  - Normally used for root cause analysis of a hazard
  - FTA is a secondary HA technique

- FTA is Like an FMEA
  - Not true
  - FMEA is bottom up single thread analysis of all item failure modes
  - FTA is a top down analysis
  - FTA only includes those failures pertinent to the top Undesired Event
FTA Criticisms

- It’s too difficult for an outside reviewer to know if a FT is complete
- The correctness of a tree cannot be verified (subjective)
- FTA cannot handle timing and sequencing
- FTA failure data makes results questionable
- FTs become too large, unwieldy and time consuming
- Different analysts sometimes produce different FTs of the same system – so one must be wrong

Most are not true
Two Equivalent FTs

Battery

Light A

Light B

Switch

Lights Fail Off

Bulb A Off

Bulb B Off

Bulb A Fails

Power Fails

Bulb B Fails

Power Fails

Switch Fails Open

Battery Fails

Lights Fail Off

Switch Fails Open

Battery Fails

Bulb A Fails

Bulb B Fails
FTA Historical Stages

- H. Watson of Bell Labs, along with A. Mearns, developed the technique for the Air Force for evaluation of the Minuteman Launch Control System, circa 1961.
- Recognized by Dave Haasl of Boeing as a significant system safety analysis tool (1963).
- The first technical papers on FTA were presented at the first System Safety Conference, held in Seattle, June 1965.
- Boeing began using FTA on the design and evaluation of commercial aircraft, circa 1966.
- Boeing developed a 12-phase fault tree simulation program, and a fault tree plotting program on a Calcomp roll plotter.
- Adopted by the Aerospace industry and Nuclear Power Industry.
- High quality FTA commercial codes developed that operates on PCs.
Reference Books

- Fault Tree Analysis Primer, C.A. Ericson, CreateSpace, 2012
Example of the Power of FTA

- No TFR Fly Up Cmd
  - No Fly Up Cmd On Primary ATF
    - No Fly Up Cmd From TFRDT
    - SCAS Lockup Prevents Fly Up
      - Relay K6 Fails Closed
        - X121
  - No Fly Up Cmd On Sec. ATF
    - Aural Fly Up Cmd Fails
      - Relay K6 Fails Closed
        - X121
    - Manual Fly Up Cmd Fails
      - Relay K6 Fails Closed
        - X121

B-1A Bomber

15 FT levels and 5 subsystems in depth.

Tree bottom shows that triple redundancy was bypassed by SPF.
--- FTA Process ---

1 - Acquire data, understand system operation.
2 - Descriptively define the problem.
3 - Define analysis ground rules and boundaries.
4 - Follow FT construction process and rules.
5 - Generate FT cut sets and probabilities.
6 - Check FT for correctness.
7 - Modify FT as found necessary.
8 - Document and apply the results.
Step 1 – Define The System

- Obtain system design information
  - Drawings, schematics, procedures, timelines
  - Failure data, exposure times
  - Logic diagrams, block diagrams, IELs

- Know and understand
  - System operation
  - System components and interfaces
  - Software design and operation
  - Hardware/software interaction
  - Maintenance operation
  - Test procedures

Guideline -- If you are unable to build block diagram of the system, your understanding may be limited.
Step 2—Define The Top Undesired Event

- **Purpose**
  - The analysis starts here, shapes entire analysis
  - Very important, must be done correctly
- **Start with basic concern**
  - Hazard, requirement, safety problem, accident/incident
- **Define the UE in a long narrative format**
- **Describe UE in short sentence**
- **Test the defined UE**
- **Determine if UE is achievable and correct**
- **Obtain concurrence on defined UE**
Example Top UE’s

- Inadvertent Weapon Unlock
- Inadvertent Weapon Release
- Incorrect Weapon Status Signals
- Failure of the MPRT Vehicle Collision Avoidance System
- Loss of All Aircraft Communication Systems
- Inadvertent Deployment of Aircraft Engine Thrust Reverser
- Offshore Oil Platform Overturns During Towing
- Loss of Auto Steer-by-wire Function
Step 3 – Establish Boundaries

- Define the analysis ground rules
- Define assumptions
- Bound the overall problem
- Obtain concurrence
- Document the ground rules, assumptions and boundaries

Boundary Factors
- System performance – areas of impact
- Size – depth and detail of analysis
- Scope of analysis – what subsystems and components to include
- System modes of operation – startup, shutdown, steady state
- System phase(s)
- Available resources (i.e., time, dollars, people)
- Resolution limit (how deep to dig)
- Establish level of analysis detail and comprehensiveness
Step 4 – Construct Fault Tree

- Follow rules and definitions of FTA
- Iterative process
- Continually check against system design
- Continually check ground rules
- Tree is developed in layers, levels and branches
Step 5 – Evaluate Fault Tree

- Qualitative Analysis
  - Generate cut sets
  - Verify correctness of cut sets
  - Evaluate cut sets for design impact

- Quantitative Analysis
  - Apply failure data to tree events
  - Compute tree probability
  - Compute importance measures
  - Evaluate probability for design impact

Generate FT results and interpret the findings
Basic Evaluation Methods

- Manual
  - possible for small/medium noncomplex trees

- Computer
  - Required for large complex trees
  - Two approaches
    - Analytical
    - Simulation

- Methods
  - Cut Set computation
    - Boolean reduction
    - Algorithms (eg, MOCUS, MICSUP)
    - Binary Decision Diagram (BDD)
  - Probability computation
    - Boolean reduction
    - Approximations
Step 6 – Validate Fault Tree

- Verify the FT is correct and accurate (Objective)
  - Check FT for errors
  - Ensure correctness
  - Best method is to check validity of every generated cut set
Step 7 – Modify Fault Tree

- Modify FT when design changes are proposed/incorporated

- Make changes in FT structure as found necessary from validation
  - Validation results
  - Risk analysis results
  - Better system knowledge

- Features that can be modified
  - Tree logic
  - Tree events
  - Event failure rates
Step 8 – Document & Apply Results

● Document the study
  ■ Customer product (in-house or external)
  ■ Historical record
  ■ May need to update FTA some day for system upgrades
  ■ May need to reference the FTA study for other projects
  ■ Adds credibility

● Apply FTA Results
  ■ Interpret results
  ■ Present the results (using the document)
  ■ Make design recommendations
  ■ Follow-up on recommendations
Summary – FTA Process

1. Define System
2. Define Top UE
3. Establish Boundaries
4. Construct Fault Tree
5. Evaluate Fault Tree
6. Validate Fault Tree
7. Modify Fault Tree
8. Doc/Apply FTA Results

SYSTEM
(Design / Data)

Fault Trees

Reports
Cut Set List
Probability
--- FTA Terms / Definitions ---

- **FT Event**
  - A basic failure event on the FT
  - A normally occurring event on the FT

- **FT Node**
  - Any gate or event on the FT

- **FT Undesired Event**
  - The hazard or problem of concern for which the root cause analysis is necessary
  - The top node or event on the FT
  - The starting point for the FT analysis
# Basic Fault Tree Symbols

## Tree Node

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Text Box</td>
<td>Contains the text for a tree node</td>
</tr>
</tbody>
</table>

## Basic Events

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="circle.png" alt="Circle" /></td>
<td>Primary Failure</td>
<td>Basic primary component failure mode</td>
</tr>
</tbody>
</table>
| ![Diamond](diamond.png) | Secondary Failure | a) Secondary component failure mode  
                         |                                      | b) Event that could be further expanded |
| ![Rectangle](rectangle.png) | Normal Event   | An event that is normally expected to occur                                  |
Basic Events (BEs)

- **Failure Event**
  - Primary Failure - basic component failure (circle)
  - Secondary Failure - failure caused by external force (diamond)

- **Normal Event**
  - An event that describes a normally expected system state
  - An operation or function that occurs as intended or designed, such as “Power Applied At Time T1”
  - The Normal event is usually either On or Off, having a probability of either 1 or 0
  - House symbol

The BE’s are where the failure rates and probabilities enter the FT
### Event Symbol Examples

**Circle**  
**Primary Failure**  
Basic inherent component failure

**Diamond**  
**Secondary Failure**  
A) Failure caused by external force

**Diamond**  
**High Level Failure**  
B) Failure that could be further developed

**House**  
**Normal Event**  
An event that would occur under normal Operation (without failure)
**Gate Events (GEs)**

- A logic operator combining input nodes

- Five basic logic operator types
  - AND, OR, Inhibit, Priority AND and Exclusive OR
  - Additional types do exist, but usually not necessary

- Represents a fault state that can be further expanded
## Gate Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="OR Gate" /></td>
<td><strong>OR Gate</strong></td>
<td>The output occurs only if at least one of the inputs occur</td>
</tr>
<tr>
<td><img src="image" alt="AND Gate" /></td>
<td><strong>AND Gate</strong></td>
<td>The output occurs only if all of the inputs occur together</td>
</tr>
<tr>
<td><img src="image" alt="Inhibit Gate" /></td>
<td><strong>Inhibit Gate</strong></td>
<td>The output occurs only if the input event occurs and the attached condition is satisfied</td>
</tr>
<tr>
<td><img src="image" alt="Exclusive OR Gate" /></td>
<td><strong>Exclusive OR Gate</strong></td>
<td>The output occurs only if at least one of the inputs occurs, but not both</td>
</tr>
<tr>
<td><img src="image" alt="Priority AND Gate" /></td>
<td><strong>Priority AND Gate</strong></td>
<td>The output occurs only if all of the inputs occur together, but in a specified sequence (input 1 must occur before 2)</td>
</tr>
</tbody>
</table>
Condition Events (CEs)

- A condition attached to a gate event
- It establishes a condition that is required to be satisfied in order for the gate event to occur

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Condition Event</td>
<td>A conditional restriction or an event probability</td>
</tr>
</tbody>
</table>
Transfer Event (TE)

- Indicates a specific tree branch (subtree)
- A pointer to a tree branch
- A Transfer only occurs at the Gate Event level
- Represented by a Triangle
- The Transfer is for several different purposes:
  - Starts a new page (for FT prints)
  - It indicates where a branch is used numerous places in the same tree, but is not repeatedly drawn (Internal Transfer)
  - It indicates an input module from a separate analysis (External Transfer)
OR Gate

Fault Tree

Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Either A or B is necessary and sufficient to cause C
- Both A and B can occur together to cause C
- Example: Light is off because light bulb fails OR power fails
**OR Gate**

- Causality passes through an OR gate
  - Inputs are identical to the output, only more specifically defined (refined) as to cause
  - The input faults are never the cause of the output fault
    - Passes the cause through
    - Not a cause-effect relationship

```plaintext
Valve Is Closed

Valve Is Closed Due To H/W Failure
Valve Is Closed Due To S/W Failure
```

Valve Is Closed

Due To H/W Failure

Due To S/W Failure
**AND Gate**

- Both A and B are necessary to cause C
- A and B must occur simultaneously
- Example: No power available because Primary power fails AND Secondary power fails

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```
AND Gate

- Specifies a causal relationship between the inputs and the output
  - Causality is created at the AND gate
  - The input faults collectively represent the cause of the output fault
  - Implies nothing about the antecedents of the input faults

```
All Site Power Is Failed

Electrical Power Is Failed
Diesel Backup Power Is Failed
Battery Backup Power Is Failed
```
**Exclusive OR Gate**

- Either A or B is necessary and sufficient to cause C
- But, both A and B cannot occur together (at same time)
- Only allow two inputs (cascade down for more ExOR inputs)
- Example: Relay is energized OR Relay is de-energized, but not both
Priority AND Gate

Fault Tree

Truth Table

- Both A and B are necessary to cause C
- But, A must occur before B
- Show priority order with inputs from left to right
- Example: Fault is not detect because Monitor fails before Computer fails
**Inhibit Gate**

- Both C and Y1 are necessary to cause D
- Y1 is a condition or a probability
- Pass through if condition is satisfied
- Example: Ignition temperature is present, given faults cause overtemp AND probability that 700 degrees is reached

Effectively an AND gate
# Transfer Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Internal Transfer" /></td>
<td>Internal Transfer</td>
<td>Indicates the start of a subtree branch, internal to present FT</td>
</tr>
<tr>
<td><img src="image" alt="External Transfer" /></td>
<td>External Transfer</td>
<td>Indicates the start of a subtree branch, external to present FT</td>
</tr>
<tr>
<td><img src="image" alt="Similar Transfer" /></td>
<td>Similar Transfer</td>
<td>Indicates the start of a subtree branch that is similar to another one, but with different hardware</td>
</tr>
</tbody>
</table>

- The transfer is a [Pointer](#) to a tree branch.
- Helps to partition trees when they become large and unwieldy.
Transfer Example

Transfer Out

Computer XR-1 Fails

Power Bus P1 Fails

Computer XR-2 Fails

Power Bus P1 Fails

Internal Transfer

External Transfer (Transfer In)

Similar Transfer
FT Diagramming (3 Ways)

Method 1 – No Internal Transfer, MOBs on same page
Internal Transfer - Equivalent FT

Method 2 – Internal Transfer, MOB on same page
Internal Transfer - Equivalent FT

Method 3 – Internal Transfer, MOB on different page
Internal Transfer - Paging

Used to indicate new page top

Page 1

Page 2

Page 3
External Transfer

FTA Study Z

- Branches A and B are not within this FT
- They are documented in FTA Study A
- Use branch top probability from Study A
- Must import if CSs are desired
Failure / Fault

- **Failure**
  - The occurrence of a *basic component failure*.
  - The result of an internal inherent failure mechanism, thereby requiring no further breakdown.
  - Example - *Resistor R77 Fails in the Open Circuit Mode*.

- **Fault**
  - The occurrence or existence of an *undesired state* for a component, subsystem or system.
  - The result of a failure or chain of faults/failures; can be further broken down.
  - The component operates correctly, except at the wrong time, because it was commanded to do so.
  - Example – The light is failed off because the switch failed open, thereby removing power.
Failure / Fault Example

Light Is Off

Fault (Command Fault)

Failure (Primary Failure)

All failures are faults, but not all faults are failures
Independent / Dependent Failure

- **Independent Failure**
  - Failure is not caused or contributed to by another event or component

- **Dependent Failure**
  - Failure is caused or contributed to by another event or component
  - A component that is caused to fail by the failure of another component
  - The two failure are directly related, and the second failure depends on the first failure occurring
  - Example - An IC fails shorted, drawing high current, resulting in resistor R77 failing open

Dependency complicates the FT mathematics
Primary Failure

- An inherent component failure mode
- Basic FT event
- A component failure that cannot be further defined at a lower level
- Example – diode inside a computer fails due to material flaw
- Symbolized by a Circle
- Has a failure rate ($\lambda$) or probability of failure

Resistor R77 fails open
Secondary Failure

- A component failure that is caused by an external force to the system
- Basic FT event
- Example – Integrated circuit fails due to external RF energy
- Important factor in Common Cause Analysis
- Symbolized by a Diamond
- Has a failure rate ($\lambda$) or probability of failure

Resistor R77 fails open from excessive RF energy
Undeveloped Failure

- A component failure that can be further defined at a lower level of detail, but is not for various reasons
  - Ground rules
  - Save analysis time and money
  - May not be a critical part of FTA
- Example – computer fails (don’t care about detail of why)
- Basic FT event
- Symbolized by a Diamond
- Has a failure rate ($\lambda$) or probability of failure

Computer CC107 fails to operate
**Command Failure**

- A fault state that is commanded by an upstream fault / failure
- Normal operation of a component, except in an inadvertent or untimely manner. The normal, but, undesired state of a component at a particular point in time
- The component operates correctly, except at the wrong time, because it was commanded to do so by upstream faults
- Example – a bridge opens (at an undesired time) because someone accidentally pushed the Bridge Open button
- Symbolized by a gate event requiring further development
FT Time Parameters

- **Mission Time**
  - The length of time the system is in operation to complete the mission
  - Most equipment is in operation during this period of time

- **Exposure Time**
  - The length of time a component is effectively exposed to failure during system operation \((P=1.0 - e^{-\lambda T})\)
  - The time assigned to equipment in FT probability calculations
  - Exposure time can be controlled by design, repair, circumvention, testing and monitoring

- **Fault Duration Time**
  - The length of time a component is effectively in the failed state
  - This state is ended by repair of the component or by system failure

\[ P = 1.0 - e^{-\lambda T} \]
System Complexity Terms

- MOE
  - A Multiple Occurring Event or failure mode that occurs more than one place in the FT
  - Also known as a redundant or repeated event
- MOB
  - A multiple occurring branch (i.e., a repeated branch)
  - A tree branch that is used in more than one place in the FT
  - All of the Basic Events within the branch would actually be MOE’s
- Branch
  - A subsection of the tree (subtree), similar to a limb on a real tree
- Module
  - A subtree or branch
  - An independent subtree that contains no outside MOE’s or MOB’s, and is not a MOB
MOE/MOB Example

- MOE is an repeated event
- MOB is a repeated branch
- All events within an MOB are effectively MOEs
Cut Set Terms

- Cut Set
  - A set of events that together cause the tree Top UE event to occur
- Min CS (MCS)
  - A CS with the minimum number of events that can still cause the top event
- Super Set
  - A CS that contains a MCS plus additional events to cause the top UE
- Critical Path
  - The highest probability CS that drives the top UE probability
- Cut Set Order
  - The number of elements in a cut set
- Cut Set Truncation
  - Removing cut sets from consideration during the FT evaluation process
  - CS’s are truncated when they exceed a specified order and/or probability
Cut Set

- A unique set of events that together cause the Top UE event to occur
- One unique root cause of the Top UE (of possibly many)
- A CS can consist of one event or multiple simultaneous events or elements

Note:
A CS element can be a:
- Failure
- Human error
- Software anomaly
- Environment condition
- Normal action
The Value of Cut Sets

- CSs identify which unique event combinations can cause the UE
- CSs provide the mechanism for probability calculations
- CSs reveal the critical and weak links in a system design
  - High probability
  - Bypass of intended safety or redundancy features

Note:
Always check all CS's against the system design to make sure they are valid and correct.
Cut Sets

AND gate means that both G & H must occur. Since they go directly to top, they comprise a CS, denoted by \{G, H\}.

Cut Set (CS)
A unique set of events that cause the Top UE to occur.
**Min CS**

A set of events that contain the minimum number of *necessary* events to cause the Top UE; it cannot be further reduced.

**Super CS**

A set of events that contain a number of events *sufficient* to cause the Top UE (ie, more than necessary as a minimum).
Min CS Example

CS1 - Night & No Headlights & Lost Keys
CS2 - Out of Gas & Dead Battery

Should be:
Night & No Headlights
Lost Keys
Out of Gas
Dead Battery

If an item can be removed from CS and top still occurs then it's not a Min CS.
**Min CS**

- A CS with the minimum number of events that can still cause the top event
- The true list of CS’s contributing to the Top
- The final CS list after removing all SCS and DupCS
- Additional CS’s are often generated, beyond the MinCS’s
  - Super Cut Sets (SCS) – result from MOE’s
  - Duplicate Cut Sets (DupCS) - result from MOE’s or AND/OR combinations
- Why eliminate SCS and DupCS?
  - Laws of Boolean algebra
  - Would make the overall tree probability slightly larger (erroneous but conservative)
Cut Sets:
A
A,B
A,B,C
A,B

Min Cut Sets:
A

SCS
DupCS, SCS

Min CS
# Alternate Gate Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Action</th>
<th>Description</th>
<th>Alternate Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Symbol]</td>
<td>Exclusive OR Gate</td>
<td>Only one of the inputs can occur, not both. Disjoint events.</td>
<td>![Alternate Symbol]</td>
</tr>
<tr>
<td>![Symbol]</td>
<td>Priority AND Gate</td>
<td>All inputs must occur, but in given order, from left to right.</td>
<td>![Alternate Symbol]</td>
</tr>
<tr>
<td>![Symbol]</td>
<td>M of N Gate</td>
<td>M of N combinations of inputs causes output to occur.</td>
<td>![Alternate Symbol] Voting Gate</td>
</tr>
</tbody>
</table>
M/N Gate Example

- M of N gate
- Also known as Voting gate
--- FT Construction Process ---

- Tree is developed in:
  - Layers
  - Levels
  - Branches

- Tree Levels:
  - **Top Level**
    - Defines the top in terms of discrete system functions that can cause the top UE
    - Shapes the overall structure of the tree
  - **Intermediate Level**
    - Defines the logical relationships between system functions and component behavior
    - Function – systems – subsystems – modules - components
  - **Bottom Level**
    - Consists of the Basic Events or component failure modes
FT Construction

I-N-S=Immediate, Necessary, Sufficient
P-S-C=Primary, Secondary, Command
S-C/S=State of the Component or System

Methodology
1) Repetitive
2) Structured
3) Methodical

Cause-Effect

Key Questions

Analysis
The 4 Basic FTA Approaches

- **Component**
  - Immediately focuses on components
  - “Shopping list” approach
  - Can overlook detailed causes

- **Subsystem**
  - Immediately emphasizes subsystems
  - Can overlook detailed causes
  - Can use Functional flow method after subsystem breakdown

- **Scenario**
  - Breaks down UE into fault scenarios before detailed design analysis
  - Sometimes necessary at FT top level for complex systems

- **Functional Flow**
  - Follows system functions (command path)
  - More structured
  - Less likely to miss detail causes
**Component Approach**

- Immediate breakdown by component
- Ignores immediate cause-effect relationships
- Tends to logically overlook things for large systems
**Subsystem Approach**

- Breakdown by subsystem
- Ignores immediate cause-effect relationships
- There can be hazard overlap between subsystems
- Tends to logically overlook things
- Eventually switch back to Functional approach

![Subsystem Diagram]

**Diagram:**
- System Fails
- Nav S/S Fails
  - A1
  - A2
- Comm S/S Fails
  - B1
  - B2
- FCS S/S Fails
  - C1
  - C2
- Prop S/S Fails
  - D1
  - D2
Scenario Approach

- Breakdown by Scenario
- Sometimes necessary to start large FTs
- Ignores immediate cause-effect relationship
- Eventually switch back to Functional approach
- Could be some overlap between subsystems
Recommended approach

- Breakdown by system function
- FTA follows system function
- Follows logical cause-effect relationship
- Has more levels and is narrower
- Less prone to miss events
- More structured and complete analysis
- Use for about 90% of applications
- FTA follows functional command path
- Structured approach
Functional Approach

- Start at UE location (E in this example)
- Follow signal flow backwards
- Take each component one at a time
Series Example

A \rightarrow B

Signal Flow

Analysis Flow

B Fails
No Output

No Input
To B

Wire Fails
Open

No Output
From B

OR

No Output
From A

No Input
To A

A Fails
No Output

OR
Series-Parallel Example

C Fails
No Output
No Input To C

A Fails
No Output
No Input To A

B Fails
No Output
No Input To B

OR

AND

Signal Flow
A
B
C

Analysis Flow

No Input To A

No Output From B

No Output From A

No Output From C
FT Construction Methodology

- Construction at each gate involves a 3 step question process:
  - Step 1 – Immediate, Necessary and Sufficient (I-N-S) ?
  - Step 2 – Primary, Secondary and Command (P-S-C) ?
  - Step 3 – State of the Component or System (S-C/S) ?

These are the 3 key questions in FTA construction
Step 1

Step 1 – *What is* Immediate, Necessary and Sufficient (I-N-S) ?

- Read the gate event wording
- Identify all *Immediate, Necessary* and *Sufficient* events to cause the Gate event
  - Immediate – do not skip past events
  - Necessary – include only what is actually necessary
  - Sufficient – do not include more than the minimum necessary
- Structure the I-N-S casual events with appropriate logic
- Mentally test the events and logic until satisfied
Step 1

C and D are Immediate
C and D are Necessary
C and D are Sufficient.

To cause Fault of E
C and D are Sufficient.
Step 2

Step 2 – What is Primary, Secondary and Command (P-S-C)?

- Read the gate event wording
- Review I-N-S events from Step 1
- Identify all **Primary**, **Secondary** and **Command** events causing the Gate event
  - Primary Fault – basic inherent component failure
  - Secondary Fault – failure caused by an external force
  - Command Fault – A fault state that is commanded by an upstream fault or failure
- Structure the P-S-C casual events with appropriate logic

If there are P-S-C inputs, then it’s an OR gate
Step 2

P = Primary Failure
S = Secondary Failure
C = Command Failure

The Command path establishes the fault flow
Step 3

Step 3 – Is it a State of the Component or System (S-C/S) fault?

- Read the gate event wording
- Identify if the Gate involves
  - a *State of the Component* fault
    - Being directly at the component level
    - Evaluating the causes of a component failure
  - a *State of the System* fault
    - Being a system level event
    - If it’s not a state of the component fault
- Structure the casual events with appropriate logic
Step 3 (continued)

- If State of the Component, then:
  - Ask “what are the P-S-C causes”
  - Generally this results in an OR gate
  - If a Command event is not involved, then this branch path is complete
Step 3 (continued)

- If State of the System, then:
  - Ask “what is I-N-S” to cause event
  - Compose the input events and logic (functional relationships)
  - This gate can be any type of gate, depending on system design
  - The input events are generally gate events

![Diagram illustrating the ARM Command Occurs with ARM Power Present and ARM Signal Present as input events.](image-url)
**P-S-C Relationship to FT Structure**

- **A - Inherent**
  - Primary Fault

- **B - External**
  - Secondary Faults

- **C - Input**
  - Command Faults

- **D - Output**
  - Undesired State

**System Component**

**Note** - Command faults follow the signal flow.
**P-S-C Example**

- **Diode** (C - Input)
- **Resistor** (A - Inherent)
- **IC** (D - Output)

**External** (B - External)

- **Resistor Fails Shorted**, **Failing IC**
- **Excessive Heat Causes Shorted Resistor**
- **Diode Shorted Causing Excessive Current**

**Primary**
- **Resistor Fails Shorted**

**Secondary**
- **Excessive Heat Causes Shorted Resistor**

**Command**
Isolate and Analyze

Analysis Views:
1) Primary - look **inward**
2) Secondary - look **outward** for incoming environmental concerns
3) Command - look **backward** at incoming signals
4) Output - look **forward** at possible undesired states that can be output
The Command path establishes the fault flow through the FT.
Example

System design -- both A and B are necessary to cause C.
Example Fault Tree

Inadv Warhead Arming

Fault In Warhead

Signal From Bomb

Fault In Bomb

Signal From Launcher

Fault In Launcher

Signal From Avionics

Fault In Avionics

Signal From Computer

Fault In Computer

Signal From W. Panel

Fault In W. Panel

Operator Error

FB-14 Bomber

Launcher

Bomb

Warhead

WARHEAD

BOMB

LAUNCHER

AVIONICS

COMPUTER

WEAPONS PANEL

Signal Flow

Analysis Flow
Construction Example

Battery

A

B

Light

Gate event under analysis

Light Fails Off

I-N-S

P, S, C

State of the Component Fault (OR gate required)

P – primary failure
S – secondary failure
C – command fault
Construction Example

Battery

Light

A

B

Light Bulb Fails

Light Receives No Current

Light Fails Off

Gate event under analysis

State of the Component Fault (OR gate required)

I-N-S

P

C

Primary Failure

Command Failure

Note – This uses P-S-C, I-N-S and S-C/S
Construction Example (continued)

Light Bulb Fails

Light Fails Off

Light Receives No Current

Gate event under analysis

State of System

Power Not Available

Ground Not Available

System Fault State

I-N-S

I-N-S

C – command fault
Construction Example (continued)

Light Fails Off

Light Bulb Fails

Light Receives No Current

Power Not Available

Ground Not Available

Ground Circuit Open

Switch A Open

Operator Opens SW A

Switch A Fails Open

Switch B Open

Operator Opens SW B

Switch B Fails Open

State of System

Battery

Light

Switch A

Switch B

Operator

Switch

Operator

Switch

Battery

Light
Construction Example (continued)

- Light Fails Off
- Light Bulb Fails
  - A
- Light Receives No Current
  - Light Fails Off

Power Not Available

- Power Source Fails
  - Battery Fails (Pwr Source)
  - Battery Fails Open (D)
- Power Path Fails
  - Wire Fails Open (E)

Ground Not Available

- Ground Circuit Open
- Ground Wire Fails Open (X2)
FT Process – Functional Flow

Note that logical Cause-Effect relationships are visible
**FT Process – Unstructured**

- The unstructured approach jumps ahead
  - Misses some important items, such as the total number of wires involved, human interaction, etc.
  - Does not depict system fault logic

**Shopping List Approach**

- Light Fails Off
  - Bulb Fails (A)
  - Switch A Fails Open (B)
  - Switch B Fails Open (C)
  - Battery Fails (D)
  - Wire Fails Open (E)

*Note that Cause-Effect relationship is not visible*
1 - Know Your System

- It is imperative to know and understand the system design and operation thoroughly
- Utilize all sources of design information
  - Drawings, procedures, block diagrams, flow diagrams, FMEAs
  - Stress analyses, failure reports, maintenance procedures
  - System interface documents
  - CONOPS
- Drawings and data must be current for current results
- Draw a Functional Diagram of the system

Rule of thumb - if you can’t construct a block diagram of system you may not understand it well enough to FT
2 - Understand The Purpose Of Your FTA

- It’s important to know why the FTA is being performed
  - To ensure adequate resources are applied
  - To ensure proper scope of analysis
  - To ensure the appropriate results are obtained

- Remember, FTA is a tool for
  - Root cause analysis
  - Identifies events contributing to an Undesired Event
  - Computes the probability of an Undesired Event
  - Measures the relative impact of a design fix
  - Logic diagrams for presentation
3 - Understand Your FT Size

- FT size impacts the entire FTA process
- As FTs grow in size many factors are affected
  - Cost (e.g., manpower)
  - Time
  - Complexity
  - Understanding
  - Traceability
  - Computation
- System factors that cause FT growth
  - System size
  - Safety criticality of system
  - System complexity
- FT factors that cause FT growth
  - MOEs and MOBs (e.g., redundancy)
  - Certain AND / OR combinations

FT size is important and has many implications
4 - *Intentionally Design Your Fault Tree*

- As a FT grows in size it is important develop an architecture and a set of rules
- The architecture lays out the overall FT design
  - Subsystem branches (for analysts and subcontractors)
  - Analyst responsibilities
- The rules provide consistent development guidelines
  - Ground rules for inclusion/exclusion (e.g., Human factors, CCFs)
  - Ground rules for depth of analysis (subsystem, LRU, component)
  - Ground rules for naming conventions (component types, MOEs)
  - Ground rules for component database

Foresight helps avoid future problems
Don't Do This! -- Plan Ahead
5 - Ensure the FT is Correct and Complete

- FT completeness is critical
  - Anything left out of the FTA skews the answer
  - The final result will only reflect what was included in the FT
  - The FTA is not complete until all root causes have been identified

- FT correctness is critical
  - If the FT is not correct the results will not be accurate

- Conduct FT peer review to ensure completeness/correctness
  - Involve other FT experts
  - Involve system designers

- Items often overlooked in FTA
  - Human error
  - Common cause failures
  - Software factors (design may have dependencies)
  - Components or subsystems considered not applicable

FT results are skewed if the FT is not complete and correct
6 - Know Your Fault Tree Tools

- Know basic FT tool capabilities
  - Construction, editing, plotting, reports, cut set evaluation
- Know FT tool user friendliness
  - Intuitive operation
  - Easy to use and remember
  - Changes are easy to implement
- Single vs. multi-phase FT
- Qualitative vs. quantitative evaluation
- Simulation vs. analytical evaluation (considerations include size, accuracy, phasing)
Tools (continued)

- Know FT tool limitations
  - Tree size (i.e., max number of events)
  - Cut set size
  - Plot size
- Understand approximations and cutoff methods, some can cause errors
- Gate probabilities could be incorrect when MOEs are involved
- Test the tool; don’t assume answers are always correct

Don’t place complete trust in a FT program
7 - Understand Your FTA Results

- Verify that the FTA goals were achieved
  - Was the analysis objective achieved
  - Are the results meaningful
  - Was FTA the right tool
  - Are adjustments necessary

- Make reasonableness tests to verify the results
  - Are the results correct
  - Look for analysis errors (logic, data, model, computer results)
  - Are CSs credible and relevant (if not revise tree)
  - Take nothing for granted from the computer
  - Test your results via manual calculations
8 - Document Your FTA

- Formally document the entire FTA
  - May need to provide to customer (product)
  - May need to defend at a later date
  - May need to modify at a later date
  - May perform a similar analysis at a later date
  - May need records for an accident/incident investigation

- Even a small analysis should be documented for posterity

- May support future questions or analyses

Documentation is essential
Documentation (continued)

- Provide complete documentation
  - Problem statement
  - Definitions
  - Ground rules
  - References
  - Comprehensive system description
  - Data and sources (drawings, failure rates, etc.)
  - FT diagrams
  - FT tree metrics
  - FT computer tool description
  - Results
  - Conclusions

Document the number of hours to perform the FTA for future estimates
Remember, it’s a “fault” tree, not a “success” tree
- Analysis of failures, faults, errors and bad designs

No magic
- Do not draw the fault tree assuming the system can be saved by a miraculous failure
- This is normally referred to as the “No Magic Rule”

No operator saves
- When constructing FT logic do not assume that operator action will save the system from fault conditions
- Only built-in safety features can be considered
- Operator errors can be considered in the FT, but not operator saves
- The system design is under investigation, not the operator performing miracles
10 - Correct Node Wording Is Important

- Be clear and precise
- Express fault event in terms of
  - Device transition
  - Input or output state
- Be very descriptive in writing event text
  - “Power supply fails” vs. “Power supply does not provide +5 VDC”
  - “Valve fails in closed position” vs. “Valve fails”
- Do not
  - Use the terms Primary, Secondary or Command
    - Thought process
    - Symbols already show it
  - Use terms Failure or Fault (if possible) – not enough information

Good node wording guides the analysis process
Wording Example

Proper wording enhances the logic process

117
11 - Follow Standard Construction Rules

- No gate-to-gate diagrams
  - Do not draw a gate without a gate node box and associated descriptive text and rectangle

- Use only one output from a node
  - Do not connect the output of a node to more than one input nodes.
  - Some analysts attempt to show redundancy this way, but it becomes cluttered and confusing.
  - Most computer codes cannot handle this situation anyway.
Construction Errors

Usually not possible with computer FT programs

Gate-to-Gate error

Single node output error
FT Construction Rules (cont’d)

- Construct the FT to most accurately reflect the system design and logic
  - Do not try to modify the tree structure to resolve an MOE.
  - Let the FT computer software handle all MOE resolutions.

- Keep single input OR gates to a minimum
  - When the words in a Node box exceed the box limit, you can create another input with a Node box directly below just to continue the words
  - Use the Notes if additional words are needed. It's okay to do but prudence is also necessary

- Use House events carefully
  - A House (Normal event) never goes into an OR gate, except in special cases, such as a multi-phase simulation FT
FT Construction Rules (cont’d)

- Do not label fault events on the tree as *Primary*, *Secondary* and *Command* failures
  - Go into detail and be descriptive. These terms are more for the thought process than the labeling process.

- When possible add traceability detail
  - Put drawing numbers and part numbers in the fault event or in the notes.
  - This provides better traceability when the tree is being reviewed or checked, or when the tree is being modified after a lengthy time period.
FT Construction Rules (cont’d)

- Operator error should be included in the analysis where appropriate
  - It is up to the analyst and the purpose/objective of the FTA as to whether the event should be included in quantitative evaluations
  - The decision needs to be documented in the analysis ground rules

- Take a second look at all tree logic structure
  - Sometimes what appears to be a simple and correct tree logic structure might actually be flawed for various reasons
    - Example -- mutually exclusive events, logic loops, etc.
  - Make sure there are no leaps or gaps in logic
  - The tree structure may need revising in these cases
12 - Provide Necessary Node Data

- Node name
- Node text
- Node type
- Basic event probability (for quantification only)

Four items are essential
13 - Apply FT Aesthetics

- When the FT structure looks good it will be better accepted
- A level FT structure looks best
  - No zig-zags
- Balance page breaks & FT structure
  - Avoid too little info on a page (i.e., 2 or 3 events)
- Always use standard FT symbols (defined in NUREG book)
- Computerized construction tools provides better graphics than manual methods

A level and balanced FT structure is easier to read
Poor Aesthetics Example

System Fails

- LRU 1 Fails
  - XX 1 Fails
  - XX 2 Fails
- LRU 2 Fails
  - WW 1 Fails
  - WW 2 Fails
- LRU 3 Fails
  - ZZ 1 Fails
  - ZZ 2 Fails
- LRU 4 Fails and the environment is bad

- Motor Fails
- Link Fails
- Comm Fails
14 - Computerized Evaluation Is Essential

- FT quantification is easy when the FT is small and simple
  - Manual calculations are easy

- FT quantification is difficult when the FT is large and complex
  - Manual quantification becomes too difficult without errors

- Hand drawn FTs typically have more errors
15 - Validate all CSs

● CSs are very important
  ■ They show where to fix system (weak design points)
  ■ They show the importance of specific components
  ■ They are necessary for most numerical calculations

● Always verify that all CSs are valid
  ■ If they are not right the FT is incorrect
16 - Perform a Numerical Reality Check

- Never completely trust the results of a computer program
  - Some algorithms may have errors
  - Proprietary approximations may not always work

- Perform a rough calculation manually to check on the computer results

- A large deviation could indicate a problem
17 - Verify All MOEs and MOBs

- Review MOEs very carefully
  - Their effect can be important - common cause, zonal analysis
  - They can cause large numerical error (or none at all)
  - They can hide or emphasize redundancy

- An MOE or MOB can be inadvertently created by erroneously using the same event name twice
18 - FTs Are Only Models

- Remember that FT’s are models
  - Perception or model of reality
  - Not 100% fidelity to exact truth
- Remember that models are approximations (generally)
  - Not necessarily 100% exact
  - Still a valuable predictor
  - Newton’s law of gravity is an approximation
- Do not represent FTA results as an exact answer
  - Use engineering judgment
  - Small number are relative (2.0x10^-8 is as good as 1.742135x10^-8)
  - Anything overlooked by the FTA skews the answer
    - Minor things left out can make results conservative (understate results)
    - Major things left out can be significant (overstate results)
19 - Understand Your Failure Data

- Failure data must be obtainable for quantitative evaluation
- Must understand failure modes, failure mechanisms and failure rates
- Data accuracy and trustworthiness must be known (confidence)
- Proven data is best
- Don’t be afraid of raw data
  - Data estimates can be used
  - Useful for rough estimate
  - Results must be understood

Even raw data provides useful results
20 – Always Provide Data Sources

- MIL-HDBK-217 Electronic Parts Predictions
- Maintenance records
- Vendor data
- Testing
- Historical databases
The human is often a key element in the system lifecycle:
- Manufacturing, assembly, installation, operation, decommissioning.

The human might be the most complex system element.

Human error includes:
- Fails to perform function (error of omission)
- Performs incorrectly
- Performs inadvertently (error of commission)
- Performs wrong function

Human error can:
- Initiate a system failure or accident
- Fail to correctly mitigate the effects of a failure (e.g., ignored warning lights)
- Exacerbate the effects of a system failure
Include Human Error in FTs

- Human error should be considered in FT model when appropriate
  - When the probability could make a difference
  - When the design needs to be modified

- Key rule – anything left out of the FT causes the results to be understated

- A poor HSI design can force the operator to commit errors
  - Mode confusion (e.g., Predator mishap)
  - Display confusion
  - Too many screens, modes and/or functions
  - GUI Widget confusion
  - Designing the system to complement the human operator
Human Reliability is Complex

- Finding human error failure data is difficult
- Rates could theoretically vary based on many factors
  - System type
  - Design
  - Human skills
  - Repetitiveness
- In general, studies show:
  - $P = 10^{-3}$ for general error
  - $P = 10^{-4}$ to $10^{-6}$ if special designs and checks are performed
22 - Node Name Length

- Short node names tend to be better than long names
  - Long names become burdensome & time consuming

- A 5 char name is easier to work with than a 24 char name
  - Typing original
  - Typing in a search
  - Storing in a database

- Random node names generated from node text tends to be more difficult to follow than shorted coded names
**Node Naming Convention**

- FT naming conventions (or coding) can be very useful
- Must maintain explicit configuration control of Event, Transfer and Gate names
  - Incorrect Event names will cause inadvertent MOE’s or none when intended
  - Incorrect Transfers names will cause use of wrong modules
  - Incorrect Gate names will cause inadvertent MOB’s or none when intended
- Most important for very large trees, not as critical for small trees
- Example: two analysts may use same diode, but each give it a different FT name
- A FT name coding scheme should be developed for the FT project
  - before the FT construction begins, planned, consistent
Sample Coding Scheme

- Use only 5 characters for a node name
- Specific characters are used to quickly identify event types
- Establish a pattern for *tree families*

<table>
<thead>
<tr>
<th>1st Char</th>
<th>Symbol Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>Gate</td>
</tr>
<tr>
<td>X</td>
<td>Circle (primary failure)</td>
</tr>
<tr>
<td>Z</td>
<td>Diamond (secondary failure)</td>
</tr>
<tr>
<td>W</td>
<td>House (normal event)</td>
</tr>
<tr>
<td>Y</td>
<td>Oval (condition event)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chars</th>
<th>Family</th>
<th>Represents</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Top level family</td>
<td>Computer System</td>
</tr>
<tr>
<td>B</td>
<td>Top level family</td>
<td>Navigation System</td>
</tr>
<tr>
<td>AA</td>
<td>Member of A</td>
<td></td>
</tr>
<tr>
<td>BB</td>
<td>Member of B</td>
<td></td>
</tr>
<tr>
<td>ABC</td>
<td>Member of AB</td>
<td></td>
</tr>
</tbody>
</table>
Coding Example

“A Tree” Family

“A Tree” Family

“B Tree” Family

“B Tree” Family

Top Level Events

139
Large FT’s necessitate FT accounting
This is a form of data control
Used in conjunction with the input roadmap
Keep accurate track of basic events:
- Name
- Text
- Failure rate
- Exposure time
- Source of data for event failure rate
- Trees where the event is used
- If it is an MOE

Generally requires a database
24 – Tools and Simplicity Help
System Design: A system is comprised of fuel, oxidizer and an ignition source. When properly applied they run an engine.

Fire scenario requires 4 things:
1) Fuel
2) Oxidizer
3) Ignition source
4) Correct flammable mixture

A fire mishap can occur if these elements should exist simultaneously due to faults. In this case the UE requires an AND gate.
Series System

- System success requires all 3 components operational.
- System failure occurs when any one component fails.
- Therefore, requires an OR gate.

A ST is often the inverse of a FT

System Success

- System Operates
  - Power Operates
  - Computer Operates
  - Monitor Operates

Success Tree (ST)

System Failure

- System Fails
  - Power Fails
  - Computer Fails
  - Monitor Fails
Parallel System

Case 1

- System success requires all 3 Motors operational.
- System failure occurs when any one Motors fails.
- Therefore, requires an OR gate.

System Success

Motor 1 Operates
Motor 2 Operates
Motor 3 Operates

System Operates

System Failure

Motor 1 Fails
Motor 2 Fails
Motor 3 Fails

System Fails
Parallel System

Case 2
• System success requires only 1 Motor operational.
• System failure occurs only when all three Motors fail.
• Therefore, requires an **AND** gate.

System Success

- Motor 1 Operates
- Motor 2 Operates
- Motor 3 Operates

System Failure

- Motor 1 Fails
- Motor 2 Fails
- Motor 3 Fails
Parallel System

Case 3
- System success requires any 2 of 3 Motors operational.
- System failure occurs when any 2 Motors fail.
- Therefore, requires combination logic.
- Note how the MOE arises in this case.

Note – In this case, If the FT is an inverted ST, the FT looks incorrect, but does work mathematically.
System Design:
A system is comprised of multiple components in series and parallel.

Therefore:
- System success requires that all series components must operate and at least one parallel component must operate successfully.
- System failure occurs if one or more series components fail, or all parallel components fail.
Sequence Parallel System

System Design:
A system is comprised of two components A and B. System success requires that both must operate successfully at the same time. System failure occurs if both fail, but only if A fails before B.

Therefore:
- Both must fail
- Sequential problem (A before B)
- The fault state logic requires a Priority AND gate.
Monitor System

System Design:
A system is comprised of two components, Monitor A and component B. Monitor A monitors the operation of B. If it detects any failure in B it takes corrective action. System success requires that B must operate successfully. System failure occurs if component B fails, which can only happen if Monitor A fails to detect a problem with B, and B subsequently fails. If A works it always corrects any failure in B or provides a warning.

This design has 2 different cases:
1. Full Monitor (full coverage)
2. Partial Monitor (partial coverage)

Coverage refers to part of subsystem that is tested. Full coverage means 100%.
Monitor System

Case 1 – Full Monitor
Monitor A monitors the operation of B, and it is designed to monitor 100% of B. In this example B is the Auto Pilot (A/P).

Therefore:
- Both must fail
- Sequential problem (A before B)
- The fault state logic requires a Priority AND gate
Monitor System

Case 2 – Partial Monitor
Monitor A monitors the operation of B, however, it is only
designed to monitor 80% of B (A/P in this case).

Therefore:
• Two problems, covered and uncovered
  segments, thus an OR gate
• Covered segment is sequential problem
  (A before B), thus the fault state logic
  requires a Priority AND gate
Standby System

System Design:
A system is comprised of two main components A and B, and a monitor M. System operation starts with component A in operation and B on standby. If A fails, then B is switched on-line and it takes over. System success requires that either A or B operate successfully. System failure occurs if both components A and B fail. Note that B can be failed if switching fails to occur.

There are three classes of Standby systems:
1. Hot Standby - powered during standby (uses operational $\lambda_O$)
2. Warm Standby - partially powered during standby ($\lambda_W < \lambda_O$)
3. Cold Standby - un-powered during standby ($\lambda_C = 0$)

The major difference between each of these modes is that the failure rate is different in the standby mode.
Standby System

Case 1 – No Assumptions
Recognizes that the Monitor is fallible, and therefore does not assume the Monitor is perfectly reliable.
Case 2 – Reliable Monitor Assumption
Assume that the Monitor is perfectly reliable.

Often modeled this way to compare with Markov analysis results, and when Case 1 is too difficult for Markov.
**Basic Reliability Equations**

- \( R = e^{-\lambda T} \)
- \( R + Q = 1 \)
- \( Q = 1 - R = 1 - e^{-\lambda T} \)
  - \( Q \approx \lambda T \) when \( \lambda T < 0.001 \) (approximation)

**Where:**
- \( R \) = Reliability or Probability of Success
- \( Q \) = Unreliability or Probability of Failure
- \( \lambda \) = component failure rate = 1 / MTBF
- MTBF = mean time between failure
- \( T \) = time interval (mission time or exposure time)

Main equation of FTA
The Effect of Failure Rate & Time

- The longer the mission (or exposure time) the higher the probability of failure
- The smaller the failure rate the lower the probability of failure

\[ P = 1 - e^{-\lambda T} \]

\[ \lambda = 1.0 \times 10^{-6} \]

The Effect of Exposure Time on Probability is Significant
**Probability**

**Union (OR Gate)**
For two events A and B, the union is the event \{A or B\} that contains all the outcomes in A, in B, or in both A and B.

**Case 1 - Disjoint Events**
\[ P = P(A) + P(B) \]

**Case 2 - Non Disjoint Events**
\[ P = P(A) + P(B) - P(A)P(B) \]

**Case 3 - Mutually Exclusive Events**
\[ P = P(A) + P(B) - 2P(A)P(B) \]

Note - Exclusive OR is not the same as Disjoint.
Intersection (AND Gate)
For two events $A$ and $B$, the intersection is the event \{A and B\} that contains the occurrence of both $A$ and $B$.

Case 1 - Independent Events
$P = P(A)P(B)$

Case 2 - Dependent Events
$P = P(A)P(B/A)$
CS Expansion Formula

\[ P = \sum \text{(singles)} - \sum \text{(pairs)} + \sum \text{(triples)} - \sum \text{(fours)} + \sum \text{(fives)} - \sum \text{(sixes)} + \ldots \]

CS \{A;\ B;\ C;\ D\}

\[ P = (P_A + P_B + P_C + P_D) \]
\[ - (P_{AB} + P_{AC} + P_{AD} + P_{BC} + P_{BD} + P_{CD}) \]
\[ + (P_{ABC} + P_{ABD} + P_{ACD} + P_{BCD}) \]
\[ - (P_{ABCD}) \]

Size and complexity of the formula depends on the total number of cut sets and MOE’s.
# Inclusion-Exclusion Approximation

CS \{A; B; C; D\}

\[ P = P_A + P_B + P_C + P_D - (P_{AB} + P_{AC} + P_{AD} + P_{BC} + P_{BD} + P_{CD}) + (P_{ABC} + P_{ABD} + P_{ACD} + P_{BCD}) - (P_{ABCD}) \]

1st Term (all singles)

2nd Term (all doubles)

3rd Term (all triples)

4th Term

---

### Graphical Representation

#### Upper Bound

#### Lower Bound

#### Exact Probability

---

<table>
<thead>
<tr>
<th>No. Of Expansion Terms</th>
<th>Exact Probability</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Min Cut Set Upper Bound Approximation

\[ P = 1 - \left[ (1 - P_{CS1})(1 - P_{CS2})(1 - P_{CS3})\ldots(1 - P_{CSN}) \right] \]

\[ CS\{A; B\} \]

\[ P = 1 - \left[ (1 - P_A)(1 - P_B) \right] \]
\[ = 1 - [1 - P_B - P_A + P_A P_B] \]
\[ = 1 - 1 + P_B + P_A - P_A P_B \]
\[ = P_A + P_B - P_A P_B \]

Equivalent to standard expansion
**FT Quantification**

Note: AND gate reduces probability

\[ P = P_A \land P_B \land P_C \land P_D \]

\[ P_T = 16 \times 10^{-24} \]

\[ P_A = 2 \times 10^{-6} \quad P_B = 2 \times 10^{-6} \quad P_C = 2 \times 10^{-6} \quad P_D = 2 \times 10^{-6} \]

Note: OR gate increases probability and math complexity

\[ P = P_A + P_B + P_C + P_D - (P_{AB} + P_{AC} + P_{AD} + P_{BC} + P_{BD} + P_{CD}) + (P_{ABC} + P_{ABD} + P_{ACD} + P_{BCD}) - (P_{ABCD}) \]

\[ P_T = 8 \times 10^{-6} \]

\[ P_A = 2 \times 10^{-6} \quad P_B = 2 \times 10^{-6} \quad P_C = 2 \times 10^{-6} \quad P_D = 2 \times 10^{-6} \]
FT Quantification

$P_A = 2 \times 10^{-6}$

$P = 4 \times 10^{-6}$

$P_A = 2 \times 10^{-6}$
$P_B = 2 \times 10^{-6}$

$P = 4 \times 10^{-6}$

$P_C = 2 \times 10^{-6}$
$P_D = 2 \times 10^{-6}$

$P = 4 \times 10^{-12}$

$P = 4 \times 10^{-12}$

$P_A = 2 \times 10^{-6}$
$P_B = 2 \times 10^{-6}$
$P_C = 2 \times 10^{-6}$
$P_D = 2 \times 10^{-6}$
Axioms of Boolean Algebra

[A1] \( ab = ba \)

[A2] \( a + b = b + a \)

[A3] \( (a + b) + c = a + (b + c) = a + b + c \)

[A4] \( (ab)c = a(bc) = abc \)

[A5] \( a(b+c) = ab + ac \)

Commutative Law

Associative Law

Distributive Law
Theorems of Boolean Algebra

[T1] \( a + 0 = a \)
[T2] \( a + 1 = 1 \)
[T3] \( a \cdot 0 = 0 \)
[T4] \( a \cdot 1 = a \)
[T5] \( a \cdot a = a \)

[T6] \( a + a = a \) \( \checkmark \) \( \) Idempotent Law
[T7] \( a \cdot \overline{a} = 0 \) \( \checkmark \)
[T8] \( a + \overline{a} = 1 \)
[T9] \( a + ab = a \) \( \checkmark \)

[T10] \( a(a + b) = a \) \( \checkmark \) \( \) Law of Absorption
[T11] \( a + \overline{ab} = a + b \) \( \checkmark \)

where \( \overline{a} = \text{not } a \)
Example

[T5] \[ a \cdot a = a \]
[T6] \[ a + a = a \]

\[ \begin{align*} \{ & \text{Idempotent Law} \end{align*} \]

Diagram: A + A → A
Example

[T9] \( a + ab = a \)
[T10] \( a(a + b) = a \)
\[ \text{Law of Absorption} \]
MOE Error Example 1

Cut Sets = A ; B ; C

\[ P = P_A + P_B + P_C \]
\[ = (2 \times 10^{-6}) + (2 \times 10^{-6}) + (2 \times 10^{-6}) \]
\[ = 6 \times 10^{-6} \quad \text{[upper bound]} \]
**MOE Error Example 2**

Cut Sets = A ; B, C

\[
P = P_A + P_B P_C
= (2 \times 10^{-6}) + (2 \times 10^{-6})(2 \times 10^{-6})
= 2 \times 10^{-6} + 4 \times 10^{-12}
= 2 \times 10^{-6} \quad \text{[upper bound]}
\]
**MOE Error Example 3**

Cut Sets = A,B ; A,C

\[
P = P_A P_B + P_A P_C = (2 \times 10^{-6})(2 \times 10^{-6}) + (2 \times 10^{-6})(2 \times 10^{-6}) = 4 \times 10^{-12} + 4 \times 10^{-12} = 8 \times 10^{-12} \quad \text{[upper bound]}
\]
--- FT Evaluation ---

**Purpose**

- Obtaining the results and conclusions from the FT

- Using the FT for its intended purpose
  - Identify root causes of UE
  - Identify critical components and paths
  - Evaluate probabilistic risk

- Using the FT to impact design
  - Identify weak links
  - Evaluate impact of changes
  - Decision making

**Evaluation Types**

- Qualitative
  - Cut Sets

- Quantitative
  - Cut Sets
  - Probability
  - Importance Measures
Methods For Finding Min CS

- Boolean reduction
- Bottom up reduction algorithms
  - MICSUP (Minimal Cut Sets Upward) algorithm
- Top down reduction algorithms
  - MOCUS (Method of Obtaining Cut Sets) algorithm
- Binary Decision Diagram (BDD)
- Min Terms method (Shannon decomposition)
- Modularization methods
- Genetic algorithms
Evaluation Trouble Makers

- Tree size
- Tree Complexity
  - Redundancy (MOEs and MOBs)
  - Large quantity of AND/OR combinations
- Exotic gates and Not logic gates
- Computer limitations
  - Speed
  - Memory size
  - Software language
- Combination of any of the above

Solutions: 1) Prune FT, 2) Truncate FT or 3) FT Simulation
**CS Truncation**

- Reduces number of CS’s when tree is too large or complex

- Order Truncation
  - Throw away all CS’s having more elements than order $N_{CO}$
  - Example – if $N_{CO}$ is 3, then CS{A, B, C, D} would be dropped

- Probability Truncation
  - Throw away all CS’s having probability smaller than $P_{CP}$
  - Example – if $P_{CP}$ is 1.0x10^{-6}, then CS(1.0x10^{-7}) would be dropped
Potential CS Truncation Errors

- With Probability truncation
  - Could discard a SPF event if the probability is below the CO

- With Order Truncation
  - Could discard a significant MinCS if all the elements have a high probability
  - If algorithm used does not completely resolve CS before discarding, could miss a MOE reduction

- With either Truncation method
  - Discarded CS’s are not included in the final probability
  - Must make sure the error is insignificant; accuracy is sacrificed
  - Circumvents any Common Cause analysis of AND gates

Do not truncate at gate level

Watch for SPFs
Min CS

- A CS with the minimum number of events that can still cause the top event
- The true list of CS’s contributing to the Top
- The final CS list after removing all SCS and DupCS
- Additional CS’s are often generated, beyond the MinCS’s
  - Super Cut Sets (SCS) – result from MOE’s
  - Duplicate Cut Sets (DupCS) - result from MOE’s or AND/OR combinations
- Why eliminate SCS and DupCS?
  - Laws of Boolean algebra
  - Would make the overall tree probability slightly larger (erroneous but conservative)
**Min CS**

Cut Sets:
- A
- A, B
- A, B, C
- A, B

SCS

DupCS, SCS

Min Cut Sets:
- A
MOCUS Algorithm

MOCUS - Method of Obtaining Cut Sets
Are these two FTs equal?

\[ G_1 = G_2 \cdot G_3 \]
\[ G_1 = (A+G_4) \cdot G_3 \]
\[ G_1 = (A+C+D) \cdot G_3 \]
\[ G_1 = (A+C+D) \cdot (B+G_5) = (A+C+D) \cdot (B+C+D) \]
\[ G_1 = AB+AC+AD+CB+CC+CD+DB+DC+DD \]
\[ G_1 = AB+AC+AD+CB+C+CD+DB+DC+D \]
\[ G_1 = AB+C+D \]

\[ G_1 = G_2 + C + D \]
\[ G_1 = AB+C+D \]
Evaluation Example

Top Down Approach (MOCUS)

G1 = G2 ● G3

A ● G3
G4 ● G3

B ● G3
C ● G3

G1 = AB + AC

C ● G5

A ● (C + AB) = AC + AAB = AC + AB
B ● (C + AB) = BC + BAB = BC + AB
C ● (C + AB) = CC + CAB = C + ABC

√ √ √
AC + AB + BC + AB + C + ABC
C + AB
Evaluation Example

Bottom Up Approach

G5 = A,B
G3 = C + G5 = C + A,B
G4 = B + C
G2 = A + G4 = A + B + C
G1 = G2 • G3
  = (A + B + C) (C + A,B)
  = A,C + A,B + B,C + A,B + C + A,B,C
  = C + A,C + B,C + A,B + A,B,C
  = C + A,B
Boolean Reductions

\[ a \cdot a = a, \quad a + a = a \]  \hspace{1cm} \text{Idempotent Law}

\[ a + ab = a, \quad a(a + b) = a \]  \hspace{1cm} \text{Law of Absorption}
MOE Reduction

[1] A

[1] A

[1] A

[1] A

[1] A, A

[1] A

[1] A

MOE Reduction


[3] B
FT Approximations vs. Markov

- **MA**
  - Small models only
  - Good numerical accuracy
  - Model is difficult to follow

- **FTA**
  - Defined, structured and rigorous methodology
  - Easy to learn, perform and follow
  - Provides root causes
  - Displays cause-consequence relationships
  - Sufficient accuracy when approximations are used

FTA is often criticized as not being accurate enough.
Series System

Description
A system is comprised of two components A and B in series. System success requires that both must operate successfully at the same time. System failure occurs if either one or both fail.

\[
P = (1 - e^{-\lambda_AT}) + (1 - e^{-\lambda_BT}) - (1 - e^{-\lambda_AT})(1 - e^{-\lambda_BT}) = 1 - e^{-\left(\lambda_A+\lambda_B\right)T}
\]

Conclusion
Both methods produce the same results (for non-repair case).
Parallel System

Description
A system is comprised of two components A and B in parallel. System success requires that either one (or both) must operate successfully. System failure occurs only if both are failed at the same time.

\[
P = (1 - e^{-\lambda_A T})(1 - e^{-\lambda_B T})
\]

\[
\frac{dP_1}{dt} = - (\lambda_A + \lambda_B)P_1 + \nu_A P_2 + \nu_B P_3
\]
\[
\frac{dP_2}{dt} = \lambda_A P_1 - (\lambda_A + \nu_A)P_2 + \nu_B P_4
\]
\[
\frac{dP_3}{dt} = \lambda_B P_1 - (\lambda_A + \nu_A)P_3 + \nu_A P_4
\]
\[
\frac{dP_4}{dt} = \lambda_B P_2 + \lambda_A P_3 - (\nu_A + \nu_B)P_4
\]

P = P_4

P = (1 - e^{-\lambda_A T})(1 - e^{-\lambda_B T})

Conclusion
Both methods produce the same results (for non-repair case).
Description
A system is comprised of two components A and B in parallel. System success requires that either one (or both) must operate successfully. System failure occurs if both fail, but only if A fails before B.

\[
P = \frac{P_A \cdot P_B}{N!}
\]

\[
P = \frac{P_A \cdot P_B}{2}
\]

\[
= \frac{(1 - e^{-\lambda_A T})(1 - e^{-\lambda_B T})}{2}
\]

Conclusion
Each method produces a different equation, but results are comparable.
Sequence Parallel System

Comparison of Results for Sequence Parallel System
Where $\lambda_A=1.0 \times 10^{-6}$ and $\lambda_B=1.0 \times 10^{-7}$

<table>
<thead>
<tr>
<th>Time (Hrs)</th>
<th>FTA</th>
<th>MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.00000E-14</td>
<td>5.00000E-14</td>
</tr>
<tr>
<td>10</td>
<td>4.99947E-12</td>
<td>4.99998E-12</td>
</tr>
<tr>
<td>100</td>
<td>4.99973E-10</td>
<td>4.99980E-10</td>
</tr>
<tr>
<td>1,000</td>
<td>4.99725E-8</td>
<td>4.99800E-8</td>
</tr>
<tr>
<td>10,000</td>
<td>4.97260E-6</td>
<td>4.98006E-6</td>
</tr>
<tr>
<td>100,000</td>
<td>4.73442E-4</td>
<td>4.80542E-4</td>
</tr>
<tr>
<td>1,000,000</td>
<td>3.00771E-2</td>
<td>3.45145E-2</td>
</tr>
</tbody>
</table>

Conclusion
Both methods produce different equations. However, for small numbers, the FTA result is a very close approximation.
--- FT Validation ---

Purpose
- Checking the FT for errors
- Verifying the FT is correct and accurate
- Checks to convince yourself that the tree is correct

Why
- Very easy to introduce errors into the FT
- FT misuse and abuse is very easy
- Helps to ensure the results are correct
- Helps to reassure the customer
- Helps to reassure management

Validation is probably one of most ignored steps in the FTA process
How Errors Are Introduced

- Analyst does not understand system
- Analyst does not fully understand FTA
- FTs can become very complex
  - Modeling a complex system design
  - FT understanding can decrease as FT size increases
- Communication errors between several FT analysts
- Errors in tree structure logic sometime occur (wrong gate selected)
- A MOE component is given the wrong name (it’s not really a MOE)
- Computer evaluation codes are erroneous
- Computer evaluation codes are used incorrectly
- Incorrect (or out of date) system data is used
  - Failure rates, drawings, design data
Methods For Validating The FT

- CS reality check
- Probability reasonableness test
- Success tree inversion
- Gate check
- Review of failure data
- Peer review
- MOE check
- Intuition check
- Logic Loop check
--- FTA Audit ---

- **Purpose** – To verify and validate a contractual FTA product

- To evaluate an existing FT for:
  - Correctness
  - Completeness
  - Thoroughness

- To determine if the results from a FTA are valid
  - Determine if the FTA contains defects
  - Avoid making decisions on incorrect analysis results
Audit vs. Validation

- A FTA audit is similar to FT validation, but not the same
- FT Audit
  - Typically performed by independent reviewer after FTA is complete
  - Auditor may not have all detailed design information or knowledge
- FT Validation
  - Typically performed by the product developer
  - Analyst has detailed design information
- Validation items that can be used for audit
  - CS reality check
  - Probability reasonableness test
  - Gate check
  - Review of failure data
  - MOE check
Audit Guidelines

- Need a basic understanding of system design (optimally)
- Evaluate FT for each potential defect category
- Question everything
  - Check with SME if possible
- If something looks funny, it probably is
- Documents audit data and results

A FT auditor:
- Must understand FT construction thoroughly
- Must be a highly experienced FT analyst
Defect Categories

- Math
- Fault logic
- Failure data
- Evaluation methods
- Completeness
  - Anything omitted
- Analysis ground rules
  - Are rules established and followed?
  - Rules on SW, HSI, CCF, exposure time, depth of analysis
- Diagramming
  - Symbol use
  - Aesthetics
## FTA Error/Defect Levels

<table>
<thead>
<tr>
<th>Error</th>
<th>Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Erroneous top probability</td>
</tr>
<tr>
<td>Medium</td>
<td>Insufficient Info; not sure if results are incorrect</td>
</tr>
<tr>
<td>Low</td>
<td>Poor FT diagram, however, results are likely correct</td>
</tr>
</tbody>
</table>
High Consequence Errors

- Gate logic error
  - AND vs. OR, logic does not correctly model system design, etc.
  - House event into an OR gate

- Omitting necessary design detail
  - Subsystems
  - Human error, SW, HSI interface design

- Cut set errors
  - Incorrect, missing, contradicting

- Mathematical errors
  - MOE resolution error, calculation error, normalizing error

- Input data errors
  - Incorrect failure rate or time

- Common FT pitfall type errors
  - Extrapolation, dependency, truncation, mutual exclusion, latency, CCF
Medium Consequence Errors

- Inadequate information errors
  - Missing text description (e.g., “Resistor fails” – open, short, tolerance?)
  - Vague text description (e.g., “Spring way too strong”)

- Missing information
  - Blank text boxes
  - Missing text boxes

- Jumping ahead in system fault path
  - Skipping fault logic steps

- Manipulations used to obtain favorable probability results

- Failure data
  - No reference sources
  - Failure rates are questionable (reasonable?)
Low Consequence Errors

- Violation of common FT rules
  - Gate to gate
  - Multiple outputs (double connects)
  - No text in boxes
  - Inputs/outputs on side of box (vice top/bottom)
  - Incorrect symbol usage

- FT Sloppiness
  - Messy diagram
  - Unreadable text (hand drawn)
  - Too small to read
Audit Checklist

- Do CSs make sense and do they cause UE
- Are all of the CSs minimal
- Are any CSs mutually exclusive
- Is the Probability reasonable (based on data and experience)
- Do Gates appear correct
- Is failure data reasonable
- Are MOEs and MOBs correct
- Does FT diagram follow basic rules
- Do all nodes have text boxes with words
- Does wording in text boxes make sense
- Does the overall fault logic seem reasonable
- Is the math correct
- Has latency been considered
- Has common cause been considered
- Has human error been considered
- Are the component exposure times correct
Audit Example #1

Premature Arming Prior to Launch
1.5 x 10

Properly Manufactured Fuze Fails Due to External Environmental Forces

Fuze Manufactured & Accepted in Armed Condition

Inspection error rate
1 x 10^-3

Anti-mal-assembly Feature
Defect Rate During Assembly
Inspection error rate
1 x 10^-3
1 x 10^-3
1.5 x 10^-2

Setback pin disengaged from rotor
Setback Spring Missing or Broken
Axial Acceleration Excess of 40G
2 Shafts (Spin Locks) Broken or Missing
Spin Rate Excess of 1100 RPM
2 Spinlocks Missing or Broken

2 Spin Lock Springs Failed
Broken
Weak
Not Installed

Set Back Pin Missing Broken

1 x 10^-3
Premature Functioning Prior to Launch

Premature Arming Prior to Launch

M55 Detonator Functioned Prematurely

M55 Detonated by Accident

Detonated By Impact

Detonated By Shock

Detonated by drop vibration

Lightening Environmental Protection Failed

Post-Launch Battery Accidentally Energized

Spin >5300 rpm

Setback >2000g

Failure of Electronic Control Processor

Failure of Electromagnetic Environmental Protection

Electromagnetic Influence

M100 Electric Detonator Functioned Prematurely

Failure of Electromagnetic Environment Protection
Premature arming after launch and prior to exit from gun

Premature arming (during launch and prior to exit from gun)

Set Back Pin functioned

R.H Detent functioned

L.H Detent functioned

Gear Train Escapement malfunctioned

Any Gear or Pinion defeated

Pallet defeated

Premature arming prior to launch

$2 \times 10^{-5}$

$1.5 \times 10^{11}$
Fuze functioned prematurely after launch and prior to exit from gun

3.644 x 10^9

M55 Detonator initiated prematurely

1.822 x 10^{-4}

Premature arming after launch and prior to exit from gun

Premature arming prior to launch

Premature arming (during launch and prior to exit from gun)

A

1.5 x 10^{44}

B

2 x 10^6

Set Back Pin functioned

R.H Detent functioned

L.H Detent functioned

Gear Train Escapement malfunctioned

2 x 10

M100 Detonator Initiated prematurely

8.22 x 10^5

1 x 10^4

Detonated by Impact

Detonated by Shock/vibration

Any Gear or Pinion defeated

Pallet defeated

1 x 10^{-5}

1 x 10^{-4}

1 x 10^{-5}

1 x 10^{-4}
Hazard to gun crew after gun exit and prior to safe separation $1.397 \times 10^{-10}$

Arming after exit and prior to safe separation

Fragmentation hazard after gun exit and prior to safe separation

Warhead function due to fuze

$3.4 \times 10^{-1}$

Firing Cap charged

Det/Delay Assembly malfunctioned

$4 \times 10^{-10}$

Firing Cap charged

Control processor malfunctioned

$1.095 \times 10^{-11}$
Audit Example #2

Source: University of Pittsburg (downloaded 4-1-09)
http://www.pitt.edu/~gartnerm/08/Incubator/Visio-FTA_Final.pdf
Audit Example #3

Fig. 2 Early design stage fault tree of an automatic transmission

Audit Guidance

- If you have received a FTA from a contractor or supplier, it’s important to obtain an independent audit of the analysis
  - Ensure the probabilities you are basing decisions on are correct
  - Require a written audit report

- Check for three defect categories
  - High, Medium and Low

- Any defects in the High category mean the FT is incorrect
Latency

- Latency refers to a latent component failure, which is a component could be failed for some time without knowledge.

- A Latent component is a component that is not checked for operability before the start of a mission. Thus, it could already be failed at the start of the mission.

- This effectively increases the component exposure time. The latent time period is the time between checks (ie, Maintenance), which can often be significantly greater than the mission time. This large exposure time can make a large impact on the probability.
**Latency**

This FT assumes both components are checked for failed state prior to flight.

- **Uncontrolled Fire Lower Aircraft Bay**
  - Fire Occurs In Lower Aircraft Bay
    - $\lambda = 1.5 \times 10^{-6}$
    - $t = 5$ Hrs
    - $P = 7.5 \times 10^{-6}$
  - Fire Detection System Fails
    - $\lambda = 1.5 \times 10^{-7}$
    - $t = 5$ Hrs
    - $P = 7.5 \times 10^{-7}$

**With latency**

- Uncontrolled Fire Lower Aircraft Bay
  - Fire Occurs In Lower Aircraft Bay
    - $\lambda = 1.5 \times 10^{-6}$
    - $t = 5$ Hrs
    - $P = 7.5 \times 10^{-6}$
  - Fire Detection System Fails
    - $\lambda = 1.5 \times 10^{-7}$
    - $T = 6,000$ Hrs
    - $P = 9.0 \times 10^{-4}$

**Note the difference**

This FT assumes the Fire Detection System cannot be checked for failed state prior to flight.
A Common Cause Failure (CCF) is a single point failure (SPF) that negates independent redundant designs.

Typical CCF sources:
- Common weakness in design redundancy
  - Example – close proximity of hydraulic lines
- The use of identical components in multiple subsystems
- Common software design
- Common manufacturing errors
- Common requirements errors
- Common production process errors
- Common maintenance errors
- Common installation errors
- Common environmental factor vulnerabilities
Design Intent

Actual Design
CCF Examples

CCF Source:
- RF energy
- Flooding
- Struck by flying engine part

CCF Source:
- SPF
Example CCF FTA

A common cause failure can cause all three Motors to fail simultaneously. This bypasses the designed redundancy. The probability of $P_{ABC}$ is the critical factor.
Example CCF FTA

Item A
Item B
Item C

System Fails
A & B Fails
A Fails
B Fails

A & C Fails
A Fails
C Fails

B & C Fails
B Fails
C Fails

ABC Fails
C AB Fails
C AC Fails
C BC Fails

C ABC Fails
A Fails
B Fails
A Fails
C Fails
B Fails
C Fails

Denotes CCF
**Interlocks**

- An interlock is usually designed into a system for one of two purposes:
  - To help prevent the inadvertent operation of a critical function
  - To stop the operation of a hazardous operation or function before a mishap occurs
- A safety interlock is a single device that is part of a larger system function; only necessary for safety, not functionality
- Its purpose is to prevent the overall system function from being performed until a specified set of safety parameters are satisfied.
- An interlock can be implemented in either hardware or software
**Interlock Example**

![Interlock Diagram]

- **Launch Power**
  - Decision Logic Interlock 1
  - Decision Logic Interlock 2
  - Decision Logic Interlock 3
  - Missile Launch

**Inadvertent Missile Launch**

- **Failures Provide Power**
  - A
  - \( P = 1.0 \times 10^{-3} \)

- **All Interlocks Are Closed**
  - \( P \approx 1.0 \times 10^{-9} \)

**Number of Interlocks**

<table>
<thead>
<tr>
<th>Number of Interlocks</th>
<th>Top Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( P \approx 1 \times 10^{-3} )</td>
</tr>
<tr>
<td>1</td>
<td>( P \approx 1 \times 10^{-6} )</td>
</tr>
<tr>
<td>2</td>
<td>( P \approx 1 \times 10^{-9} )</td>
</tr>
<tr>
<td>3</td>
<td>( P \approx 1 \times 10^{-12} )</td>
</tr>
</tbody>
</table>
Dependency

- An Independent event is an event that **is not** influenced or caused by another event.
- A Dependent event is an event that **is** influenced or caused by another event.
- Dependencies complicate FT math considerably:
  - Conditional probability
  - Requires Markov analysis for accuracy
  - However, FT approximations are quite accurate.
- Sometimes dependencies are difficult to identify:
  - A Secondary failure may or may **not** be the cause of a dependent failure.
  - If A causes B, then in this case Prob(B/A) should be more likely than independent Prob(B).
  - Secondary RF energy may cause a transistor to fail, but they are “typically” considered independent (the approximation is accurate enough).
Dependency vs. Independency

Loss Of Two Engines

- Engines 1 & 2 Fail Independently
  - Engine 1 Fails
  - Engine 2 Fails

- Engine 1 Fails & Engine 2 Shutdown
- Engine 2 Fails & Engine 1 Shutdown
- Engine 1 Fails And Causes Engine 2 Fail
- Engine 2 Fails And Causes Engine 1 Fail

- Engine 1 Fails Due To Overload
- Cross Wiring Fault
- Human Error Fault

Dependent Failure? What if 2 had not failed?

Independent Failure
Software level is based upon the contribution of software to potential failure conditions as determined by the system safety assessment process (SSAP).

The software level implies that the level of effort required to show compliance with certification requirements varies with the failure condition category.

<table>
<thead>
<tr>
<th>Level</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Software whose anomalous behavior, as shown by the SSAP, would cause or contribute to a failure of system function resulting in a catastrophic failure condition for the aircraft</td>
</tr>
<tr>
<td>B</td>
<td>Software whose anomalous behavior, as shown by the SSAP, would cause or contribute to a failure of system function resulting in a hazardous/severe-major failure condition of the aircraft</td>
</tr>
<tr>
<td>C</td>
<td>Software whose anomalous behavior, as shown by the SSAP, would cause or contribute to a failure of system function resulting in a major failure condition for the aircraft</td>
</tr>
<tr>
<td>D</td>
<td>Software whose anomalous behavior, as shown by the SSAP, would cause or contribute to a failure of system function resulting in a minor failure condition for the aircraft</td>
</tr>
<tr>
<td>E</td>
<td>Software whose anomalous behavior, as shown by the SSAP, would cause or contribute to a failure of function with no effect on aircraft operational capability or pilot workload</td>
</tr>
</tbody>
</table>
FT for SIL Level

Loss Of All Attitude Data

Secondary Display Blanks
  - HW3 Faults
  - SW2 Faults

Both Primary Displays Blank
  - Captain’s PFD Blanks
    - HW1 Faults
    - SW1 Faults
  - F/O’s PFD Blanks
    - HW2 Faults
    - SW1 Faults

Common SW

DO-178B
A – Catastrophic
B – Severe
C – Major
D – Minor
E – No safety effect

PFD=Primary Flight Display

Reduced to B because of AND gate above

Not reduced to C because of common SW
--- FTA Software ---

Basic Tasks of FTA Software

- Construction
  - creating FT
  - editing FT
- Evaluation
  - cut sets
  - probability
  - importance measures
- Plotting / Printing
  - plotter
  - printer
- Reports
  - data
  - results

Diagram:

```
Computer
| FT Editor |
| Display Generator |
| Plot Generator |
| CS Engine |
| Probability Engine |
| Report Generator |

User

Display
Prints/Plots
Reports
```
## Properties of FT Software

<table>
<thead>
<tr>
<th>Function</th>
<th>Required</th>
<th>Desired</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construction (create, edit, copy, paste)</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Generate Min CS</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Generate probabilities (top, gates)</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Employ CS cutoff methods (order, probability)</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Numerical accuracy</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Generate reports</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Detect tree logic loops</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Data export capability (tree structure, failure data)</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Graphic export capability (BMP, JPG)</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Tree Pagination for prints</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>User select print size</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Program fit on a floppy disk</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Notes on FT print/plot</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Find feature</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Undo feature</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
## Properties (continued)

<table>
<thead>
<tr>
<th>Required</th>
<th>Desired</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>● Employ MOEs / MOBs</td>
<td>X</td>
</tr>
<tr>
<td>● Correctly resolve MOEs / MOBs</td>
<td>X</td>
</tr>
<tr>
<td>● Global data change (failure rates, exposure time)</td>
<td>X</td>
</tr>
<tr>
<td>● Print/plot selected pages or all pages</td>
<td>X</td>
</tr>
<tr>
<td>● Unrestricted FT tree size</td>
<td>X</td>
</tr>
<tr>
<td>● Automatic naming of gates and events when created</td>
<td>X</td>
</tr>
<tr>
<td>● Capability to resolve large complex FTs</td>
<td>X</td>
</tr>
<tr>
<td>● User friendly (intuitive commands)</td>
<td>X</td>
</tr>
<tr>
<td>● Print/plot results visually aesthetic</td>
<td>X</td>
</tr>
<tr>
<td>● Verification of mathematical methods and accuracy</td>
<td>X</td>
</tr>
<tr>
<td>● Open data file structure</td>
<td>X</td>
</tr>
</tbody>
</table>
# Commercial FT Software

<table>
<thead>
<tr>
<th>Software</th>
<th>Website/Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAFTA</td>
<td>CAFTA sales 650-384-5693 (part of EPRI)</td>
</tr>
<tr>
<td>Fault Tree+</td>
<td><a href="http://www.Isograph.com">www.Isograph.com</a></td>
</tr>
<tr>
<td>Item</td>
<td><a href="http://www.itemsoft.com">www.itemsoft.com</a></td>
</tr>
<tr>
<td>Relex</td>
<td><a href="http://www.relexsoftware.com">www.relexsoftware.com</a></td>
</tr>
<tr>
<td>FaulTrEase</td>
<td><a href="http://www.chempute.com/faultrea.htm">http://www.chempute.com/faultrea.htm</a></td>
</tr>
<tr>
<td>Risk Spectrum FT</td>
<td><a href="http://www.relcon.com">http://www.relcon.com</a></td>
</tr>
<tr>
<td>Shade Tree</td>
<td><a href="http://www.qrainc.com">http://www.qrainc.com</a></td>
</tr>
<tr>
<td>Tree-Master</td>
<td><a href="http://www.mgtsciences.com">http://www.mgtsciences.com</a></td>
</tr>
<tr>
<td>FTA Pro</td>
<td><a href="http://www.dyadem.com">http://www.dyadem.com</a></td>
</tr>
<tr>
<td>OpenFTA (free source)</td>
<td><a href="http://www.openfta.com">http://www.openfta.com</a></td>
</tr>
<tr>
<td>RAM Commander</td>
<td><a href="http://www.aldservice.com">http://www.aldservice.com</a></td>
</tr>
<tr>
<td>Saphire (free?)</td>
<td><a href="https://saphire.inl.gov/faq.cfm">https://saphire.inl.gov/faq.cfm</a></td>
</tr>
</tbody>
</table>
Purchase Considerations

- Many commercial FT codes are available -- some good and some?
- Know FT tool limits and capabilities
  - Tree size, Cut set size, Print size, numerical accuracy
  - Easy to use (without tech support)
  - Single phase, multi-phase
- Understand algorithms
  - Some codes have errors in approximations and cutoff methods
  - Gate probability calculations must resolve MOEs correctly
- Test the tool; don’t assume answers are always correct
- Consider
  - Price
  - Lease vs. Ownership
  - User flexibility – networks, stations, individuals
  - Maintenance
  - Lease / Buy
  - User friendliness
  - Training
Mission Calculation Errors

Example 1

- Shortcut error (calculate for 1 hr, then multiple by 100 hrs for trade study)

P≠(Prob of 1 Hr Mission) x (100 Hrs)

Incorrect

1 Hr Mission

\[ P_{TOP} = P_A \times P_B \times P_C \]
\[ = \lambda_A T \times \lambda_B T \times \lambda_C T \]
\[ = (1 \times 10^{-6})(1) \times (1 \times 10^{-6})(1) \times \]
\[ (1 \times 10^{-6})(1) \]
\[ = 1 \times 10^{-18} \]

Adjusted for 100 Hr Mission

\[ P_{TOP} = P_{TOP} \times 100 \text{ Hrs} \]
\[ = (1 \times 10^{-18})(100) \]
\[ = 1 \times 10^{-16} \]

Correct

100 Hr Mission

\[ P_{TOP} = P_A \times P_B \times P_C \]
\[ = \lambda_A T \times \lambda_B T \times \lambda_C T \]
\[ = (1 \times 10^{-6})(100) \times (1 \times 10^{-6})(100) \times \]
\[ (1 \times 10^{-6})(100) \]
\[ = 1 \times 10^{-12} \]
**Example 2**

- **Shortcut error (multiply $$\lambda$$’s)**

$$P \neq (\lambda_{\text{TOTAL}}) \times \text{(Mission Time)}$$

**Incorrect**

<table>
<thead>
<tr>
<th>100 Hr Mission</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$P_{\text{TOP}} = \lambda_{\text{TOP}} \cdot T$$</td>
</tr>
<tr>
<td>= $$(\lambda_A \cdot \lambda_B \cdot \lambda_C) \cdot (T)$$</td>
</tr>
<tr>
<td>= $$(1 \times 10^{-6}) \cdot (1 \times 10^{-6}) \cdot (1 \times 10^{-6})(T)$$</td>
</tr>
<tr>
<td>= $$1 \times 10^{-18} \ (T)$$</td>
</tr>
<tr>
<td>= $$1 \times 10^{-18} \ (100)$$</td>
</tr>
<tr>
<td>= $$1 \times 10^{-16}$$</td>
</tr>
</tbody>
</table>

**Correct**

<table>
<thead>
<tr>
<th>100 Hr Mission</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$P_{\text{TOP}} = \lambda_A T \cdot \lambda_B T \cdot \lambda_C T$$</td>
</tr>
<tr>
<td>= $$(\lambda_A \cdot \lambda_B \cdot \lambda_C) \cdot (T^3)$$</td>
</tr>
<tr>
<td>= $$(1 \times 10^{-6})(1 \times 10^{-6})(1 \times 10^{-6})(10^6)$$</td>
</tr>
<tr>
<td>= $$1 \times 10^{-12}$$</td>
</tr>
</tbody>
</table>
Ignoring Critical Items

Some important items are often ignored or overlooked
• Latency
• Human error
• CCFs

Human Error (HE) (SPF)

Safety Critical Function (SCF)

HE & CCF can have significant impact
Summary

- There are good FTs, mediocre FTs and bad FTs
- Strive to construct good FTs
- Understanding FT rules and intentionally designing a FT helps to make a quality product
- Don’t just spit out a mediocre FT in order to meet a deadline or CDRL
- It’s easy to visually inspect the quality of a FT
  - A good FT analyst can tell how much effort and credibility someone put into their FT just through a visual inspection
- In order to perform a quality FTA the analyst must thoroughly understand the system and the unique process of FTA