Introduction To Fault Tree Analysis

Clifton A. Ericson II
Design Safety Solutions LLC
cliftonericson@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

--- FTA Introduction ---

Fault Tree Handbook with Aerospace Applications (updated NUREG-0492), 2002

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.

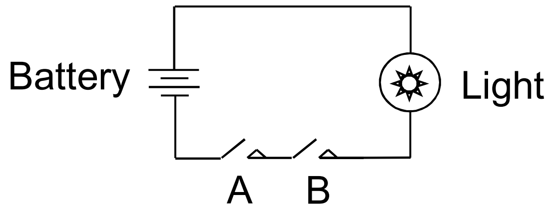
Credibility

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

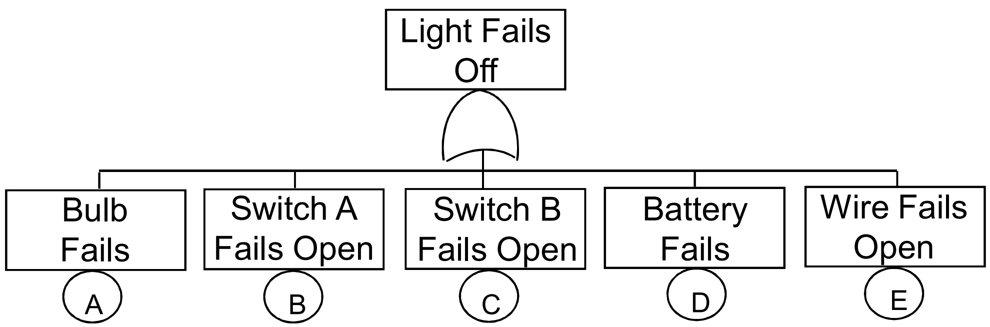
Example FT

System



System Undesired Event: *Light Fails Off*

FT Model

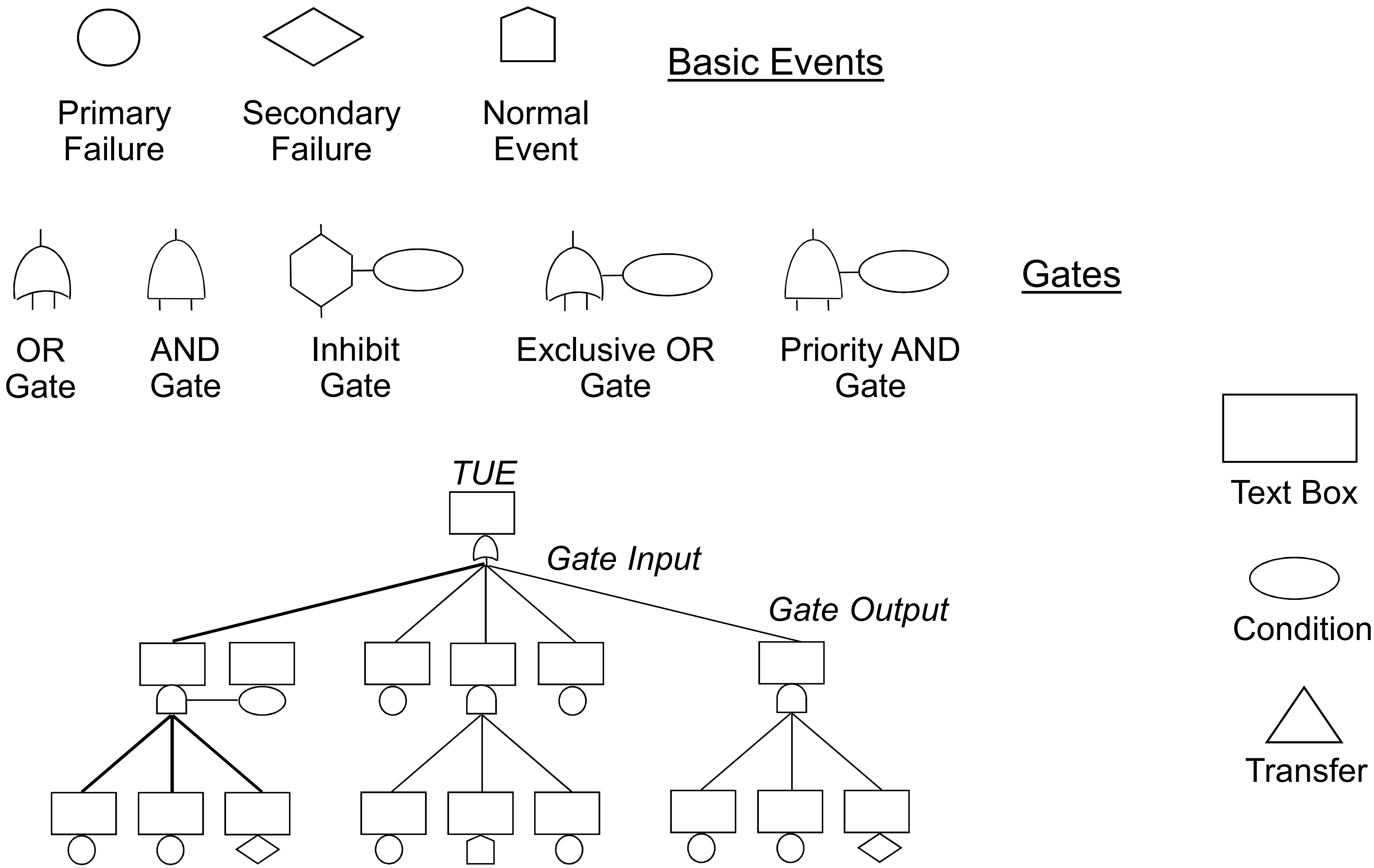


Cut Sets

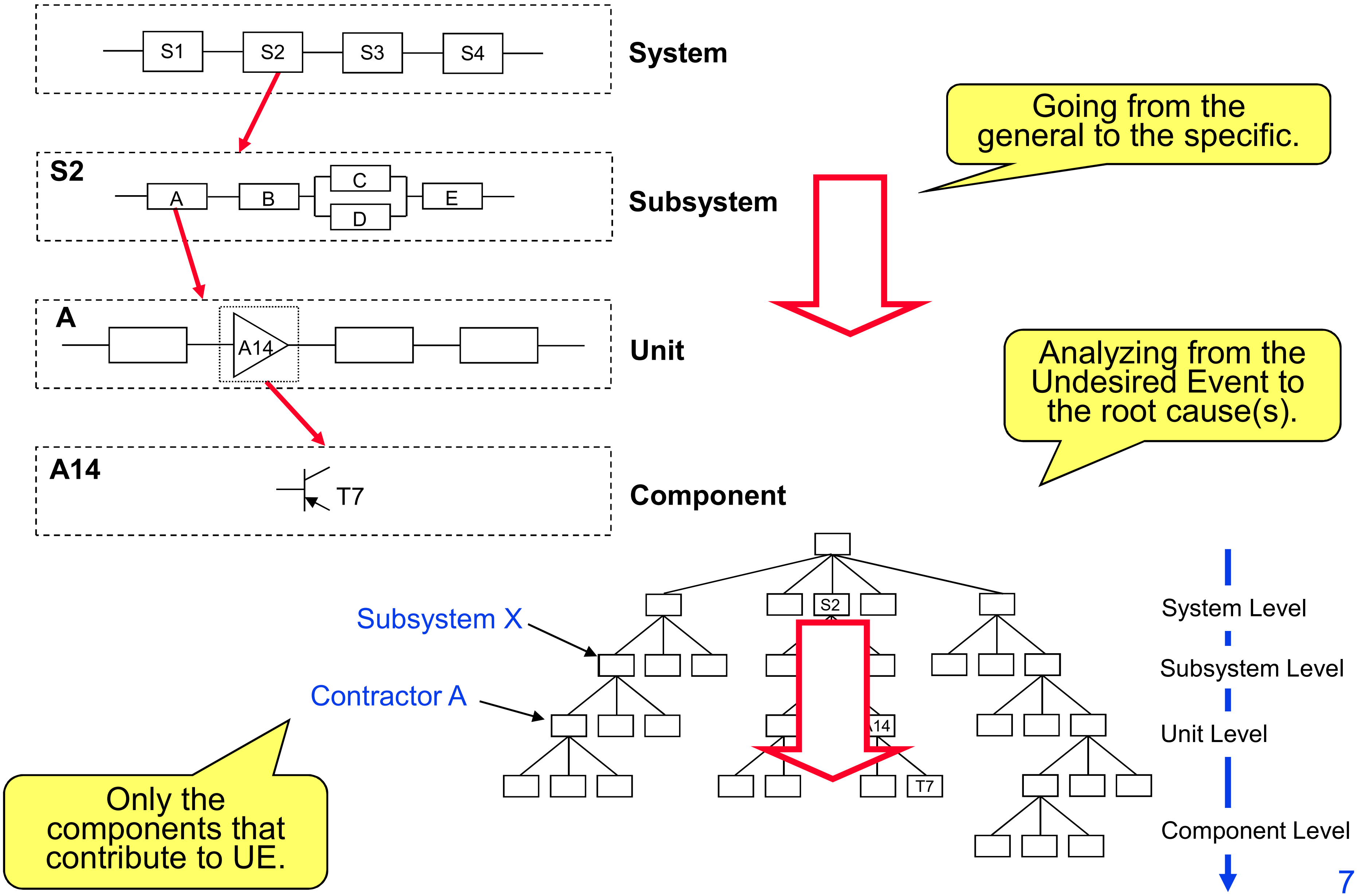
Event combinations that can cause Top Undesired Event to occur

CS	Probability
A	$P_A=1.0x10^{-6}$
B	$P_B=1.0x10^{-7}$
C	$P_C=1.0x10^{-7}$
D	$P_D=1.0x10^{-6}$
E	$P_E=1.0x10^{-9}$

FT Building Blocks



FTA – Deductive Approach



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

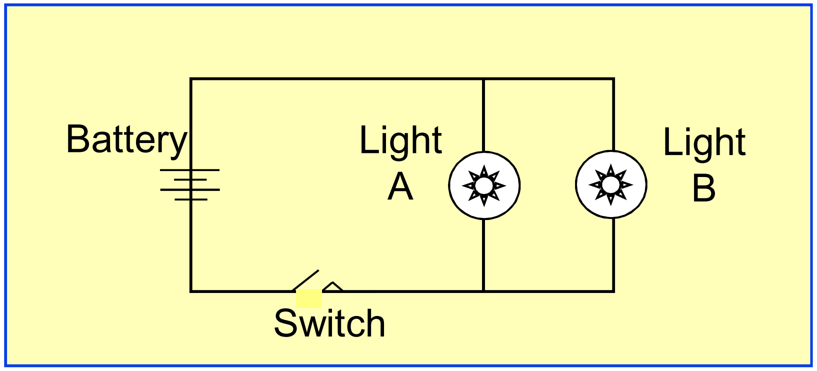
- FTA is Like an FMEA
  - Not true
  - FMEA is bottom up single thread analysis of all item failure modes
  - FTA is a top down analysis
  - FTA only includes those failures pertinent to the top Undesired Event

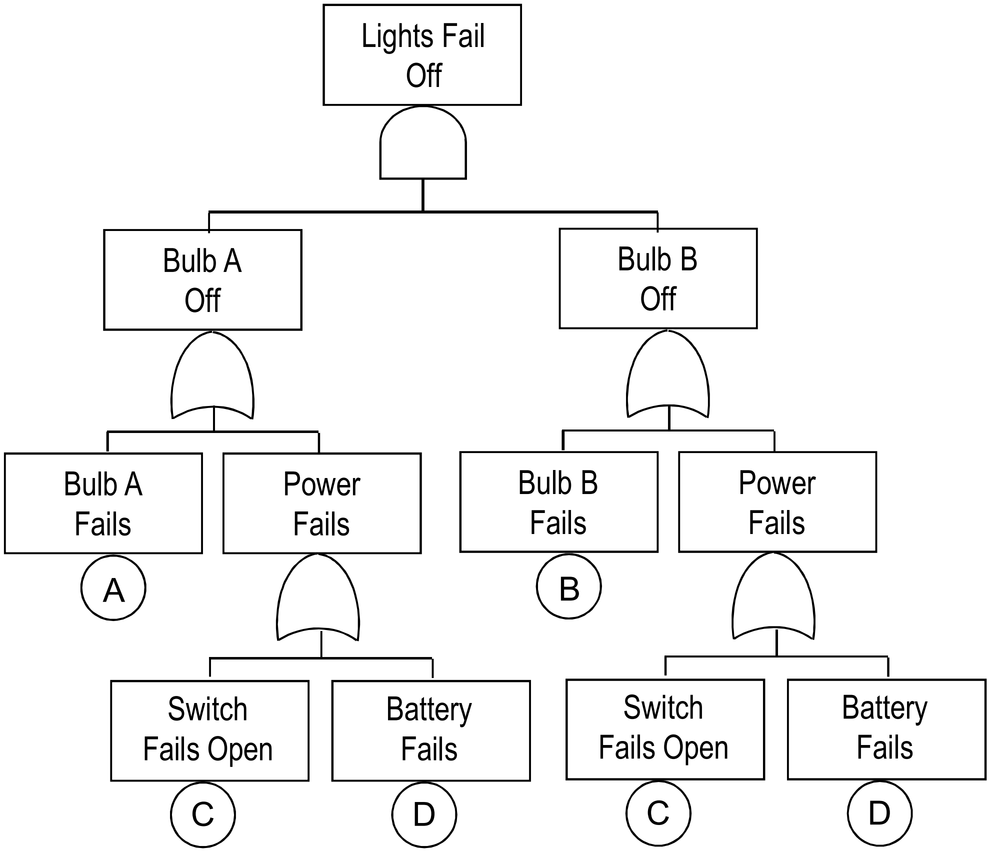
FTA Criticisms

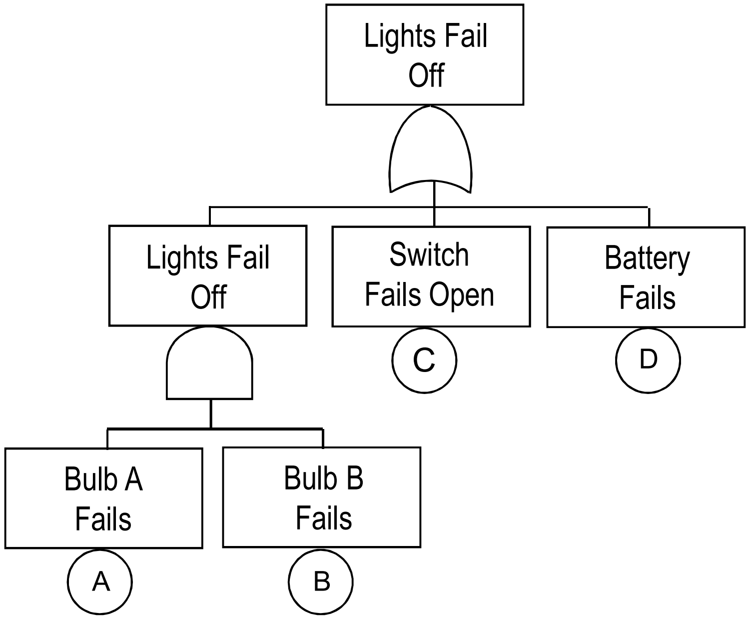
- It's too difficult for an outside reviewer to know if a FT is complete
- The correctness of a tree cannot be verified (subjective)
- FTA cannot handle timing and sequencing
- FTA failure data makes results questionable
- FTs become too large, unwieldy and time consuming
- Different analysts sometimes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

Most are not true

Two Equivalent FTs







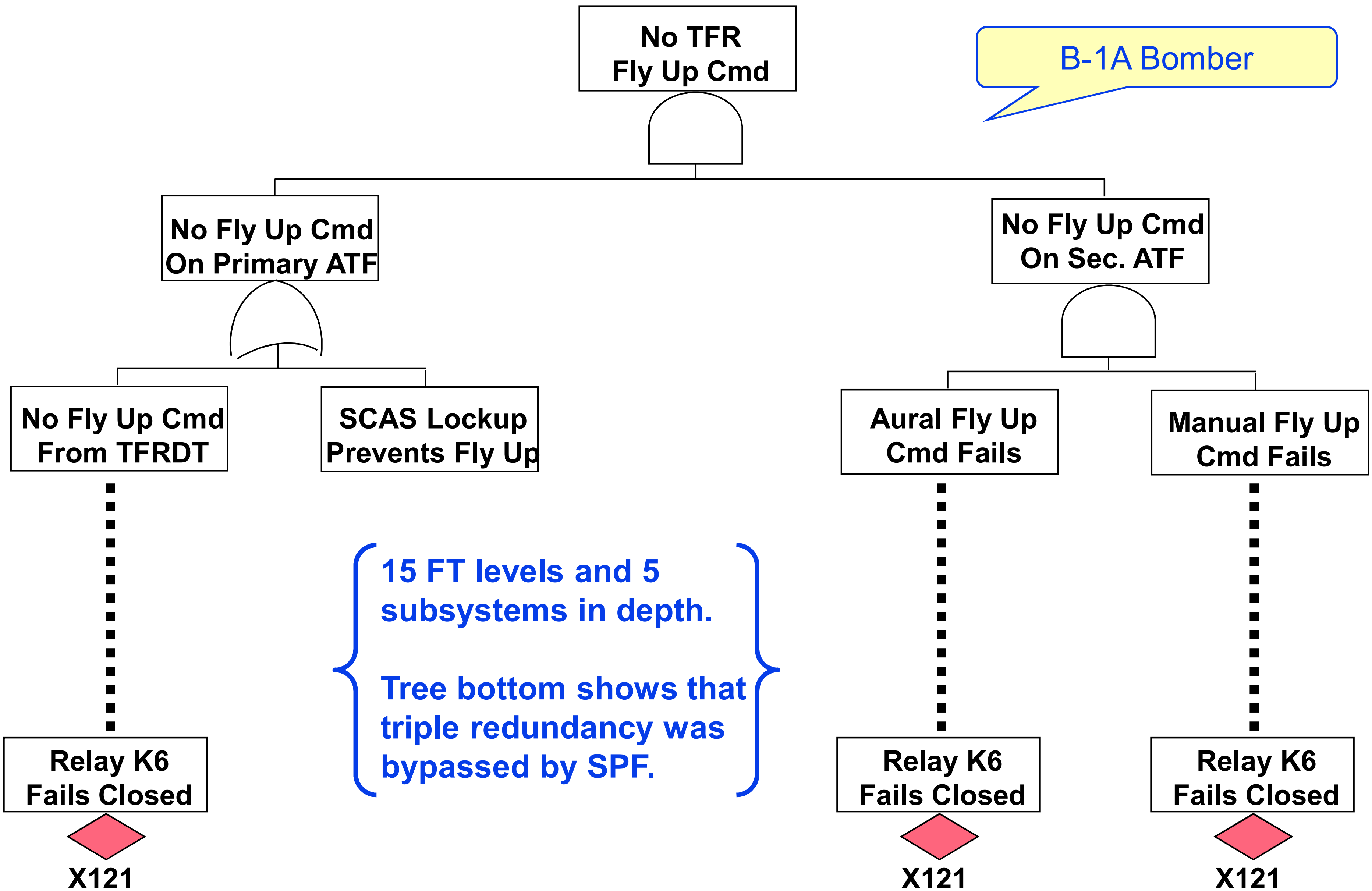
FTA Historical Stages

- 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)
- First major use when applied by Boeing on the entire Minuteman system for safety evaluation (1964 – 1967, 1968-1999)
- 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

- Reliability and Fault Tree Analysis, Conference On Reliability And Fault Tree Analysis; UC Berkeley; SIAM Pub, R. E. Barlow & J. B. Fussell & N. D. Singpurwalla, 1975.
- NUREG-0492, Fault Tree Handbook, N. H. Roberts, W. E. Vesely, D. F. Haasl & F. F. Goldberg, 1981.
- IEC 1025, Fault Tree Analysis, International Electrotechnical Commission, 1990.
- Reliability and Risk Assessment, Longman Scientific & Technical, 1993, J. D. Andrews & T. R. Moss, 1993.
- Probabilistic Risk Assessment and Management for Engineers and Scientists, E. J. Henley & H. Kumamoto, IEEE Press (2nd edition), 1996.
- NASA (no number), Fault Tree Handbook with Aerospace Applications, August 2002.
- NASA (no number), Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners, August 2002.
- Hazard Analysis Techniques for System Safety, C. A. Ericson, John Wiley & Sons, 2005, Chapter 11.
- Fault Tree Analysis Primer, C.A. Ericson, CreateSpace, 2012

Example of the Power of FTA



--- FTA Process ---

Define the System	1 - Acquire data, understand system operation.
Define Top Undesired Event	2 - Descriptively define the problem.
Establish Boundaries	3 - Define analysis ground rules and boundaries.
Construct Fault Tree	4 - Follow FT construction process and rules.
Evaluate Fault Tree	5 - Generate FT cut sets and probabilities.
Validate Fault Tree	6 - Check FT for correctness.
Modify Fault Tree	7 - Modify FT as found necessary.
Document & Apply Results	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

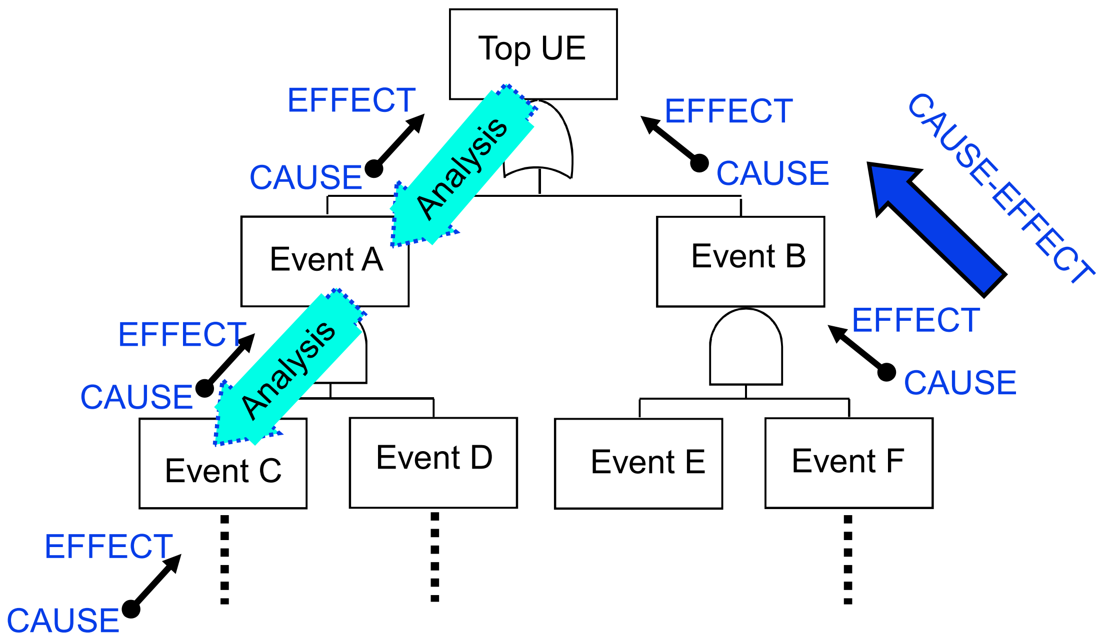
- Define the analysis ground rules
- Define assumptions
- Bound the overall problem
- Obtain concurrence
- Document the ground rules, assumptions and boundaries

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

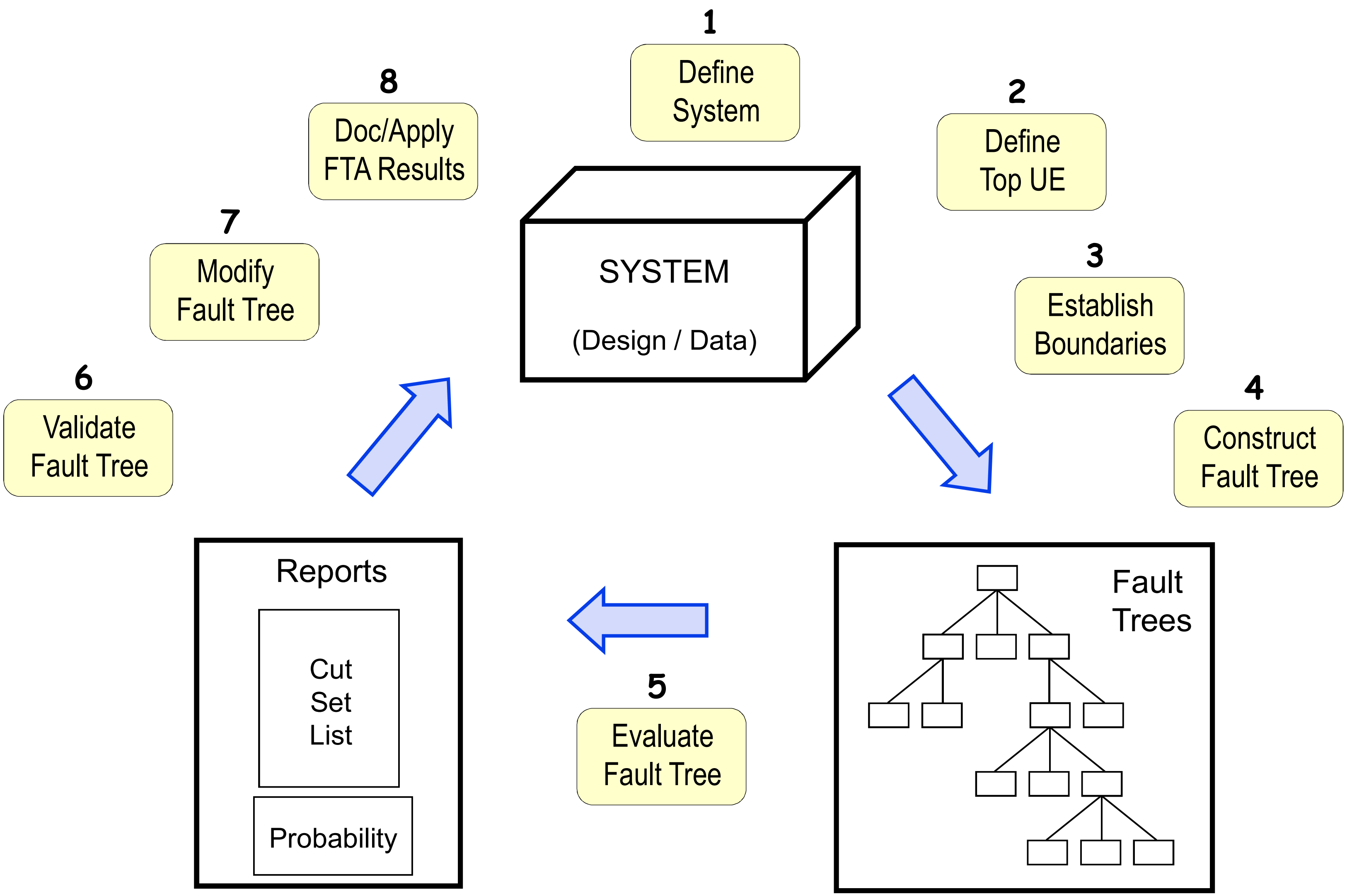
- Modify FT when design changes are proposed/incorporated
- Make changes in FT structure as found necessary from validation
  - Validation results
  - Risk analysis results
  - Better system knowledge
- Features that can be modified
  - Tree logic
  - Tree events
  - Event failure rates

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



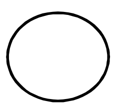
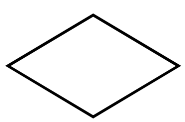
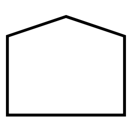
--- FTA Terms / Definitions ---

- 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

Symbol	Action	Description
	Text Box	Contains the text for a tree node

Tree Node

Symbol	Action	Description
	Primary Failure	Basic primary component failure mode
	Secondary Failure	a) Secondary component failure mode b) Event that could be further expanded
	Normal Event	An event that is normally expected to occur

Basic Events

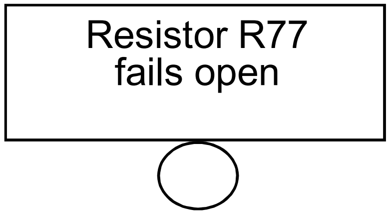
Basic Events (BEs)

- 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

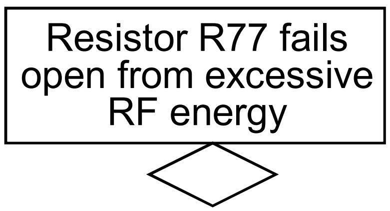
The BE's are where the failure rates and probabilities enter the FT

Event Symbol Examples



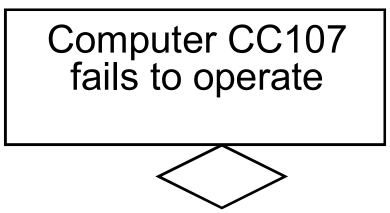
Circle
Primary Failure

Basic inherent component failure



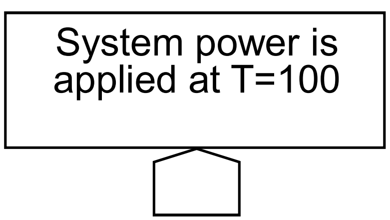
Diamond
Secondary Failure

A) Failure caused by external force



Diamond
High Level Failure

B) Failure that could be further developed



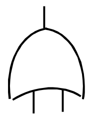
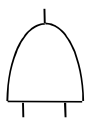
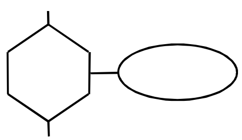
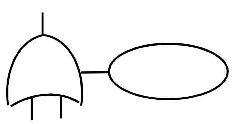
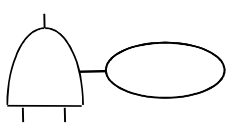
House
Normal Event

An event that would occur under normal Operation (without failure)

Gate Events (GEs)

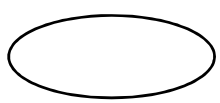
- A logic operator combining input nodes
- Five basic logic operator types
  - AND, OR, Inhibit, Priority AND and Exclusive OR
  - Additional types do exist, but usually not necessary
- Represents a fault state that can be further expanded

Gate Symbols

Symbol	Action	Description
	OR Gate	The output occurs only if at least one of the inputs occur
	AND Gate	The output occurs only if all of the inputs occur together
	Inhibit Gate	The output occurs only if the input event occurs and the attached condition is satisfied
	Exclusive OR Gate	The output occurs only if at least one of the inputs occurs, but not both
	Priority AND Gate	The output occurs only if all of the inputs occur together, but in a specified sequence (input 1 must occur before 2)

Condition Events (CEs)

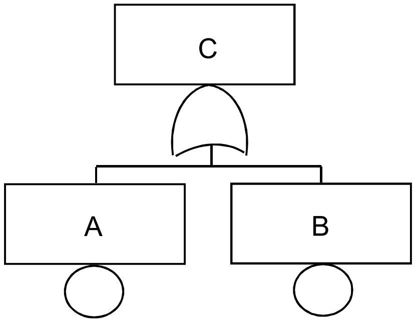
- A condition attached to a gate event
- It establishes a condition that is required to be satisfies in order for the gate event to occur

Symbol	Action	Description
	Condition Event	A conditional restriction or an event probability

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

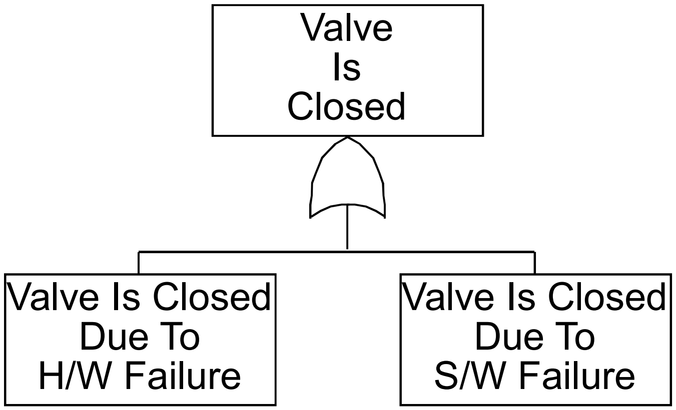
A	B	C
0	0	0
1	0	1
0	1	1
1	1	1

Truth Table

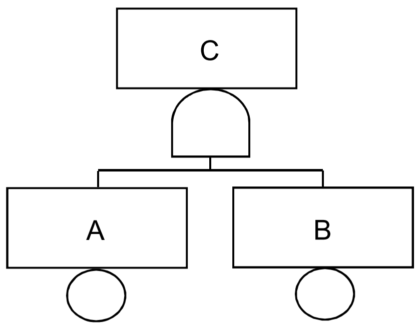
- Either A or B is necessary and sufficient to cause C
- Both A and B can occur together to cause C
- Example: Light is off because light bulb fails OR power fails

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



AND Gate



Fault Tree

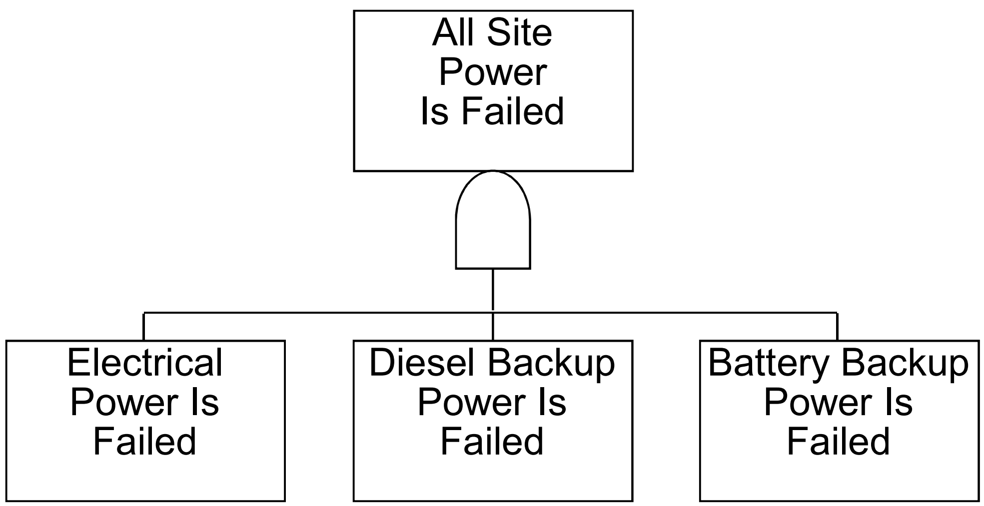
A	B	C
0	0	0
1	0	0
0	1	0
1	1	1

Truth Table

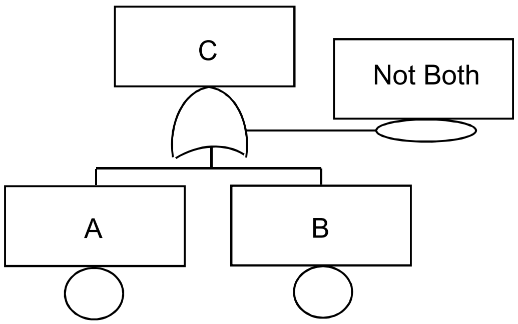
- Both A and B are necessary to cause C
- A and B must occur simultaneously
- Example: No power available because Primary power fails AND Secondary power fails

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



Exclusive OR Gate



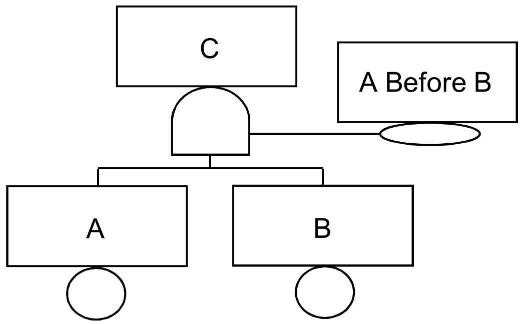
Fault Tree

A	B	C
0	0	0
1	0	1
0	1	1
1	1	0

Truth Table

- 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



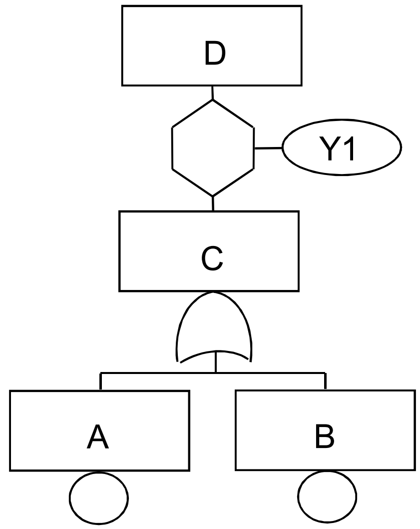
Fault Tree

A	B	C
0	0	0
1	0	0
0	1	0
1	1	1

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

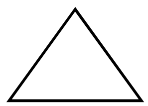
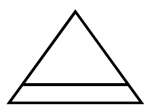
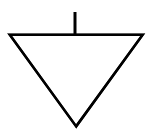
Inhibit Gate



Effectively an AND gate

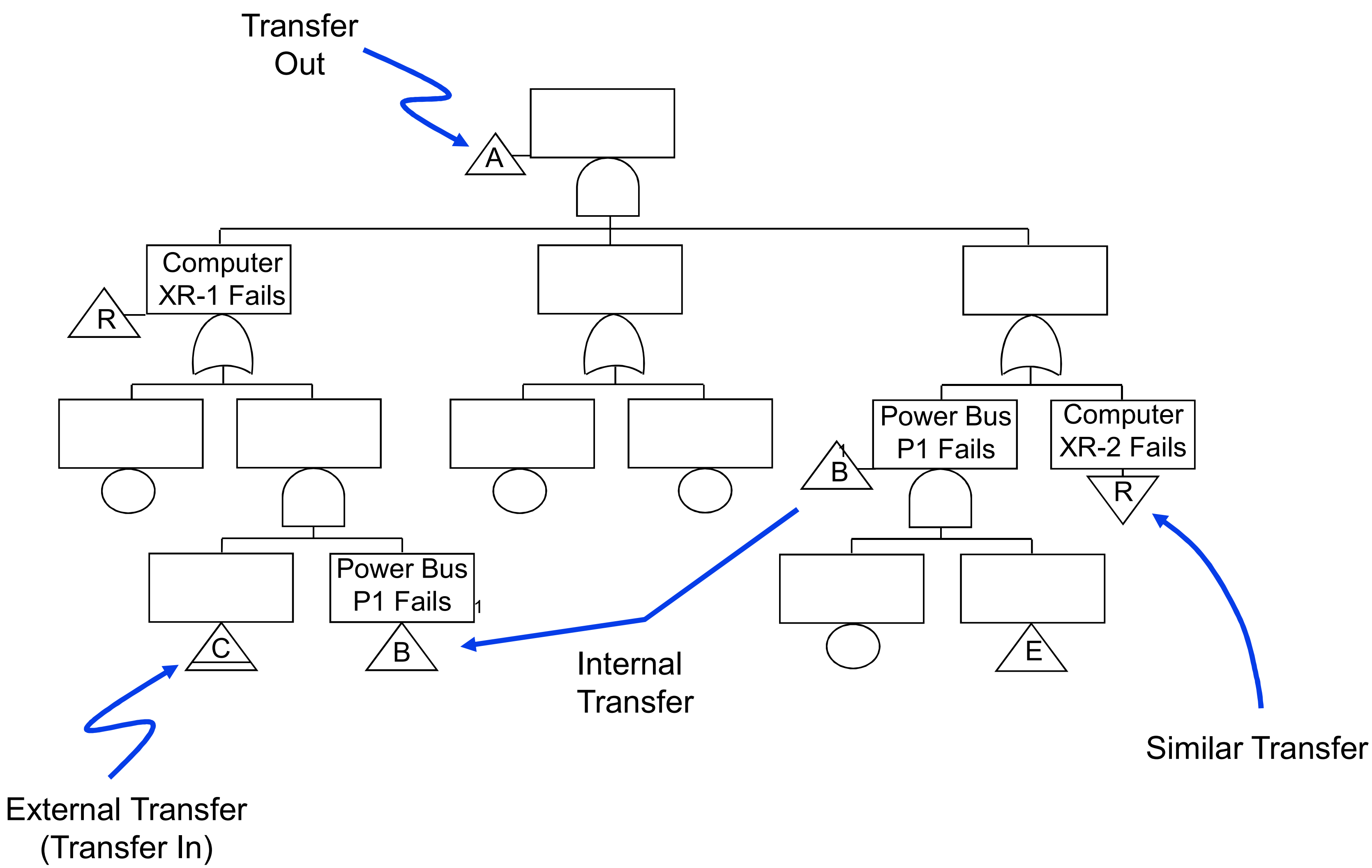
- Both C and Y1 are necessary to cause D
- Y1 is a condition or a probability
- Pass through if condition is satisfied
- Example: Ignition temperature is present, given faults cause overtemp AND probability that 700 degrees is reached

Transfer Symbols

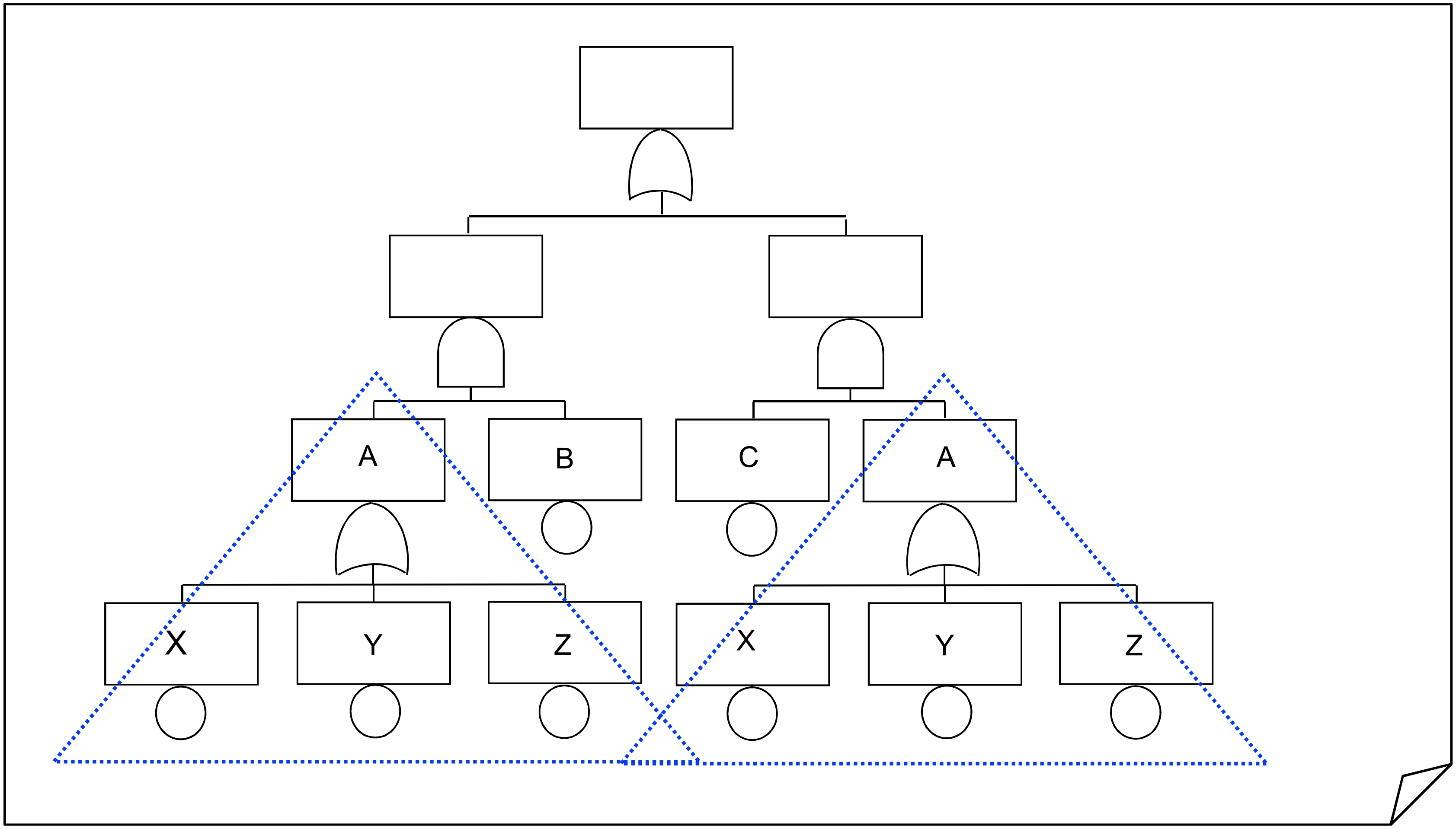
Symbol	Action	Description
	Internal Transfer	Indicates the start of a subtree branch, internal to present FT
	External Transfer	Indicates the start of a subtree branch, external to present FT
	Similar Transfer	Indicates the start of a subtree branch that is similar to another one, but with different hardware

- The transfer is a Pointer to a tree branch.
- Helps to partition trees when they become large and unwieldy.

Transfer Example

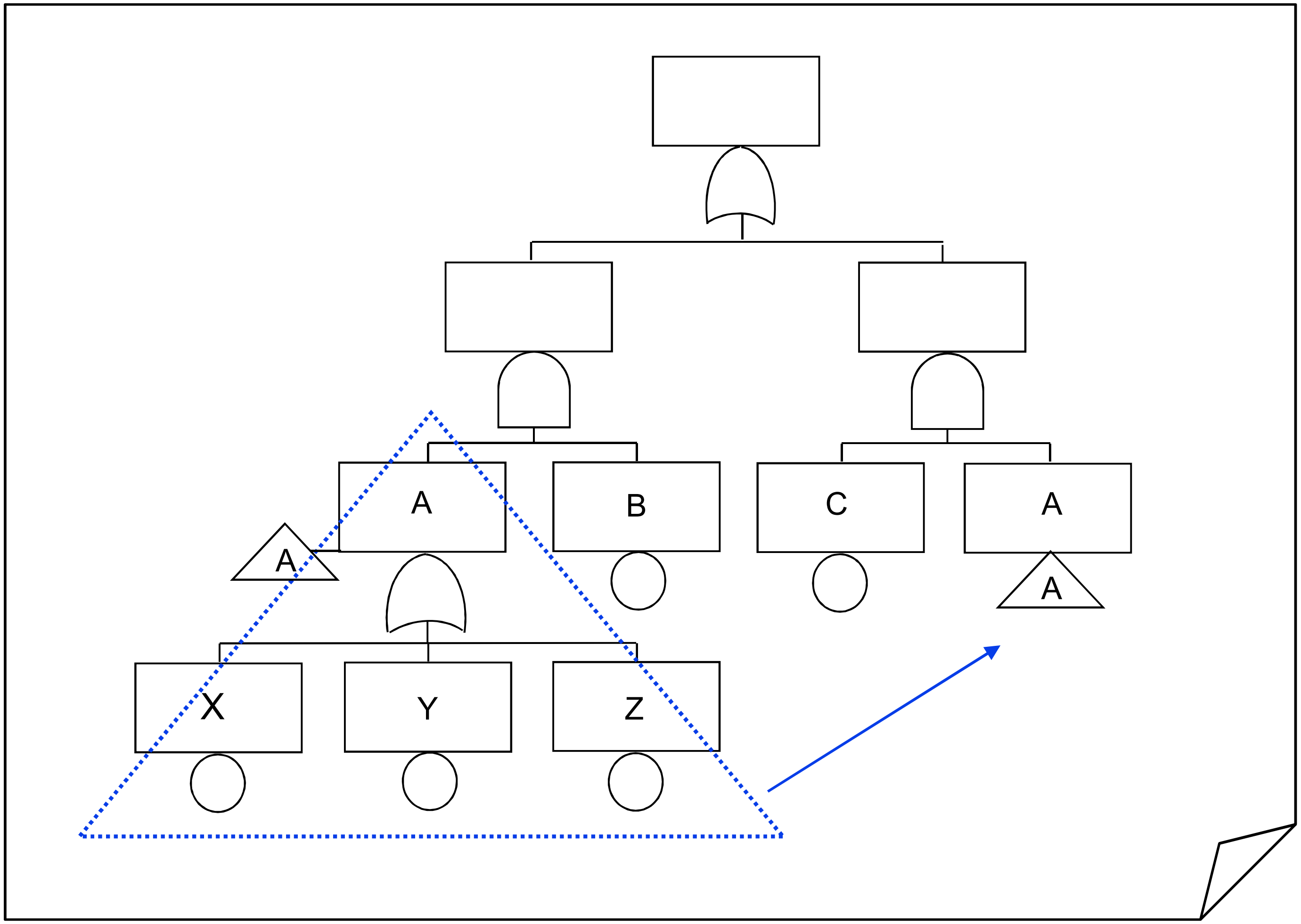


FT Diagramming (3 Ways)



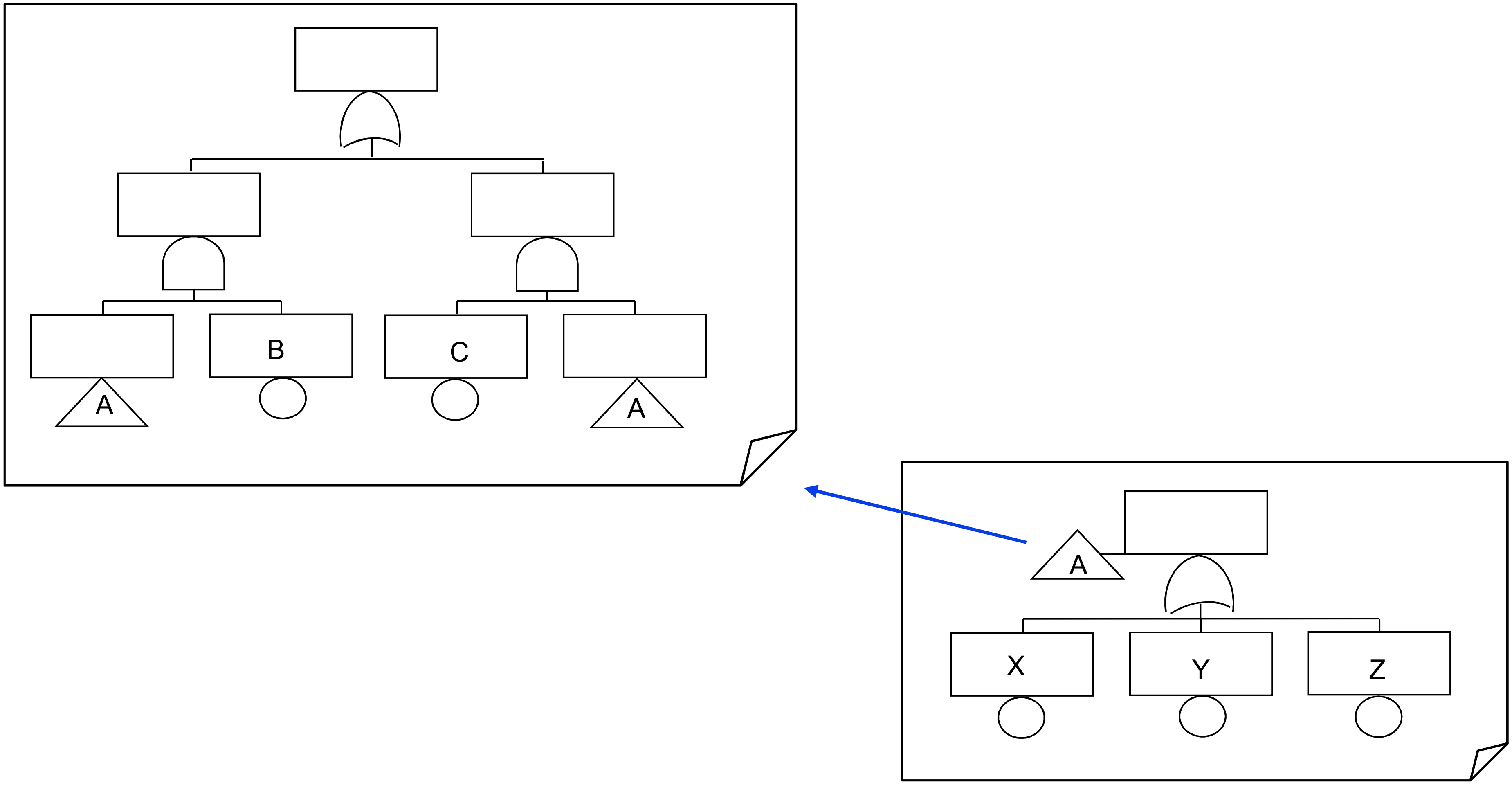
Method 1 – No Internal Transfer, MOBs on same page

Internal Transfer - Equivalent FT



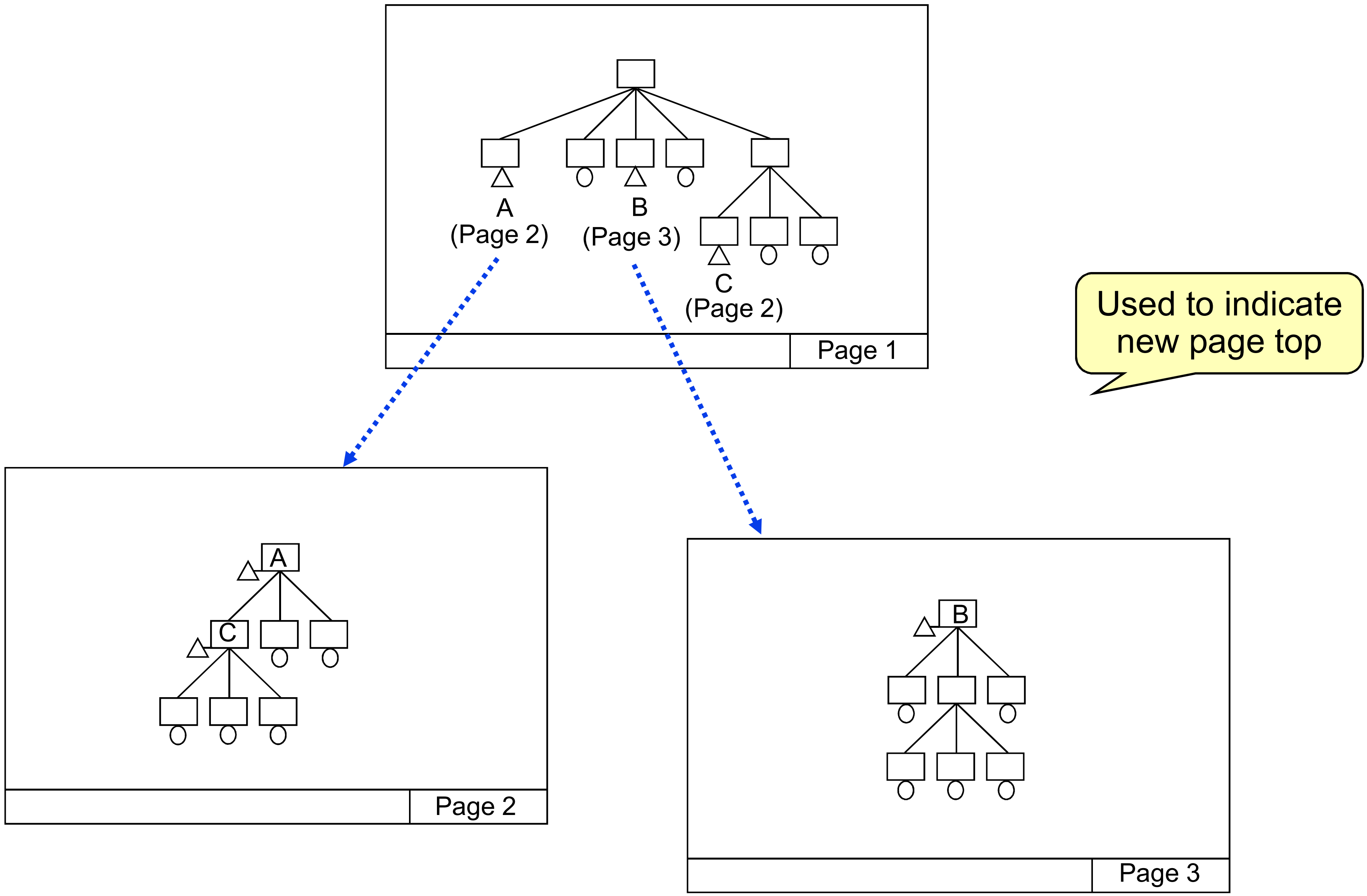
Method 2 – Internal Transfer, MOB on same page

Internal Transfer - Equivalent FT

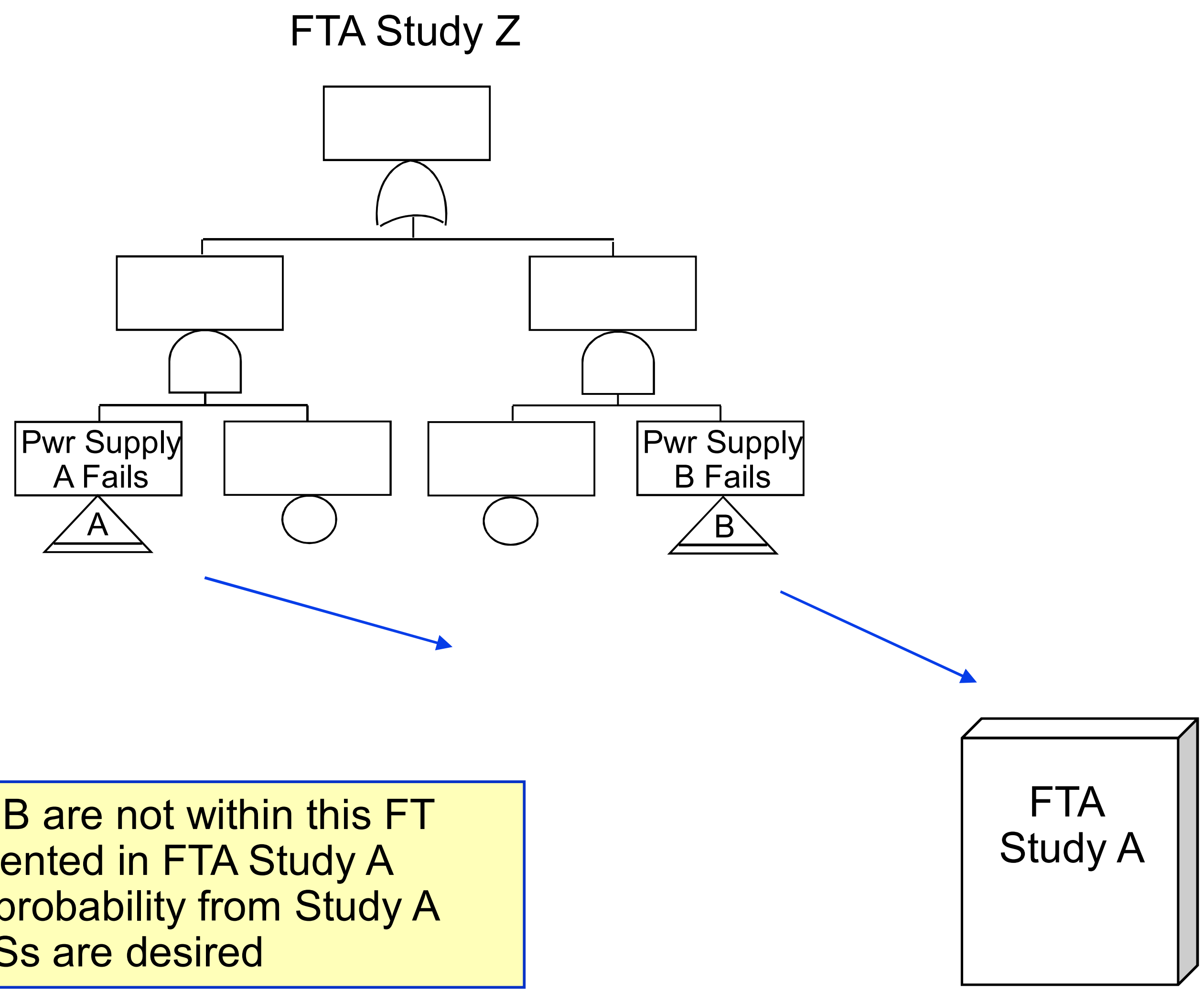


Method 3 – Internal Transfer, MOB on different page

Internal Transfer - Paging



External Transfer

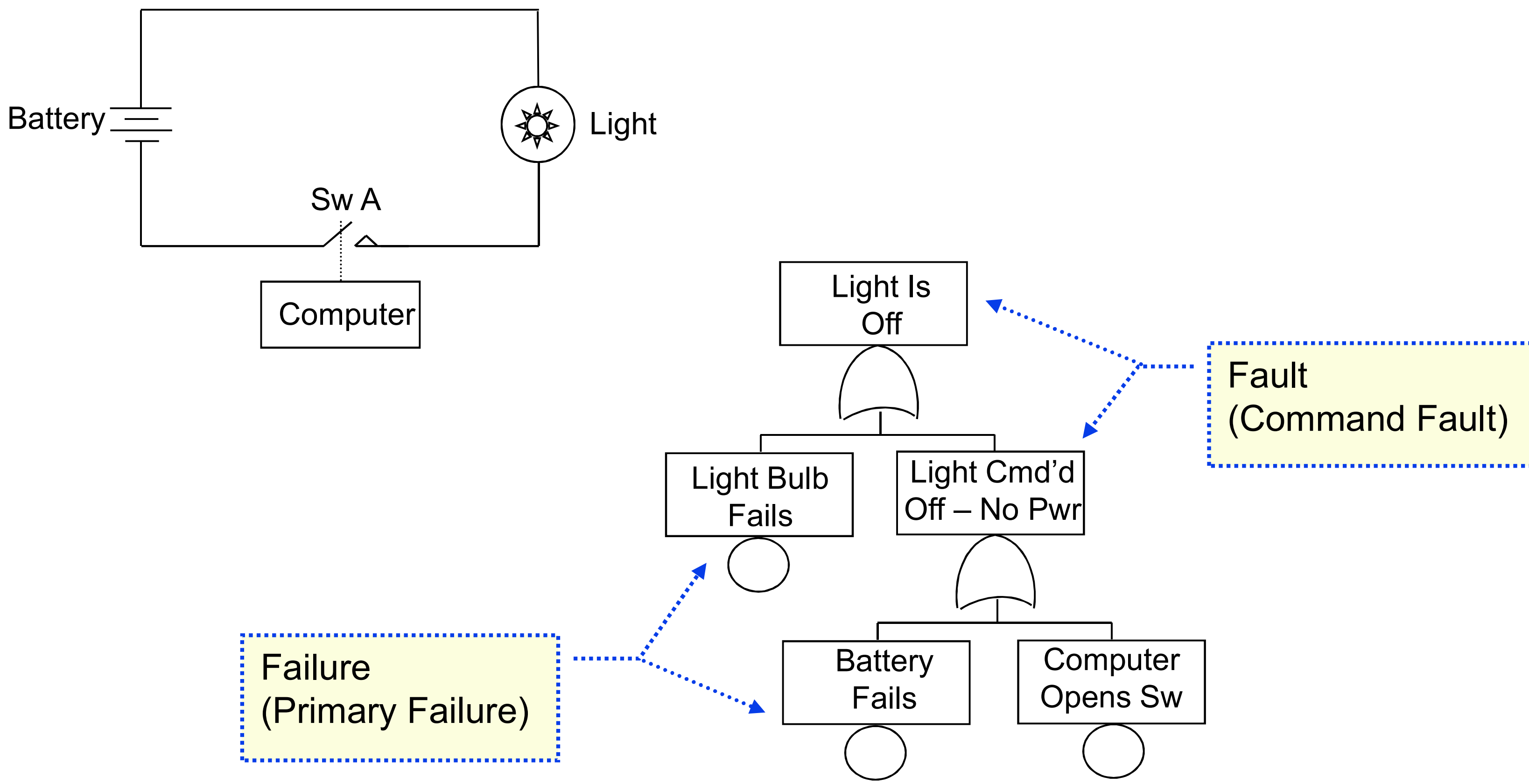


- Branches A and B are not within this FT
- They are documented in FTA Study A
- Use branch top probability from Study A
- Must import if CSs are desired

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



All failures are faults, but not all faults are failures

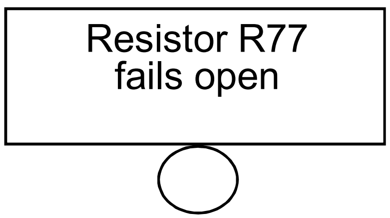
Independent / Dependent Failure

- Independent Failure
  - Failure <u>is not</u> caused or contributed to by another event or component
- Dependent Failure
  - Failure <u>is</u> 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

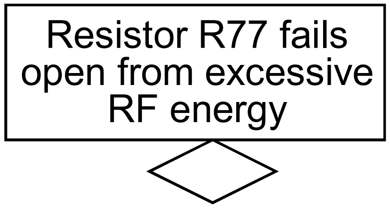
Primary Failure

- 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 materiel flaw
- Symbolized by a Circle
- Has a failure rate (λ) or probability of failure



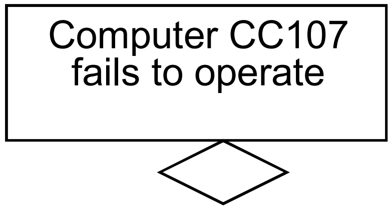
Secondary Failure

- 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 (λ) or probability of failure



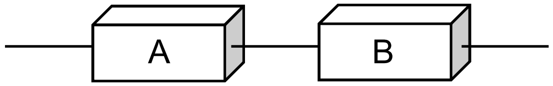
Undeveloped Failure

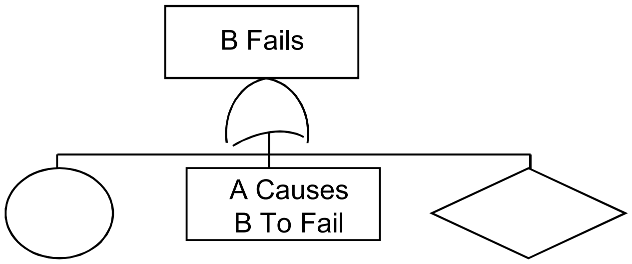
- A component failure that can be further defined at a lower level of detail, but is not for various reasons
  - Ground rules
  - Save analysis time and money
  - May not be a critical part of FTA
- Example – computer fails (don't care about detail of why)
- Basic FT event
- Symbolized by a Diamond
-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



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

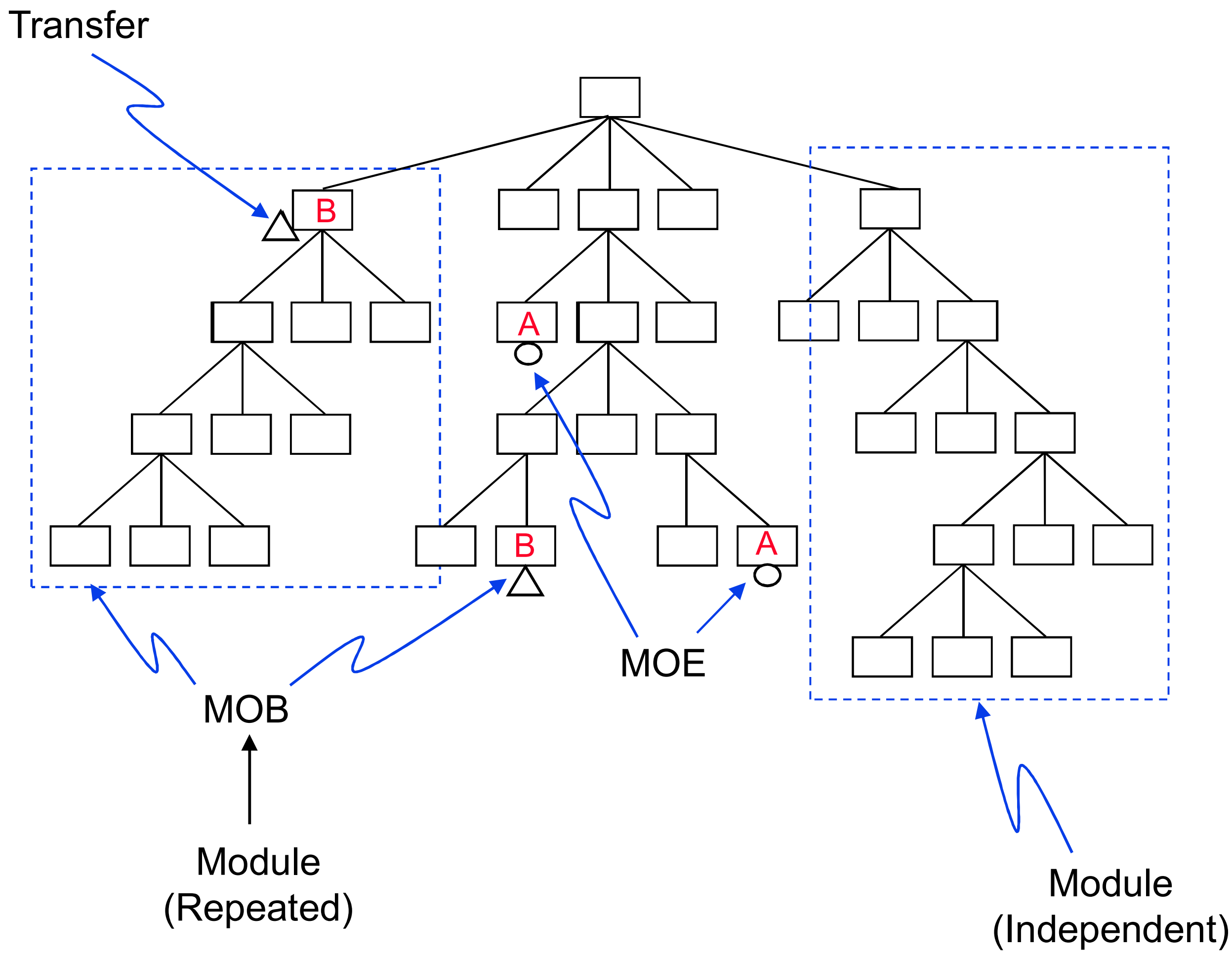
Time

$$P = 1.0 - e^{-\lambda T}$$

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

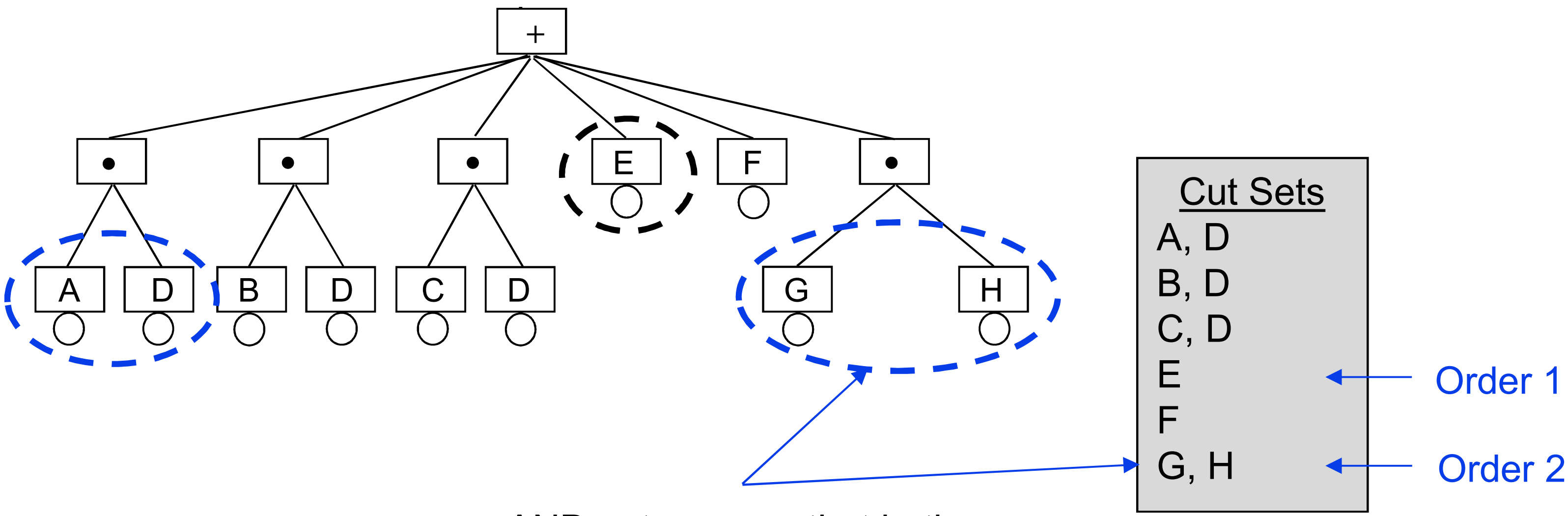
Note:
A CS element can be a:
- Failure
- Human error
- Software anomaly
- Environment condition
- Normal action

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.

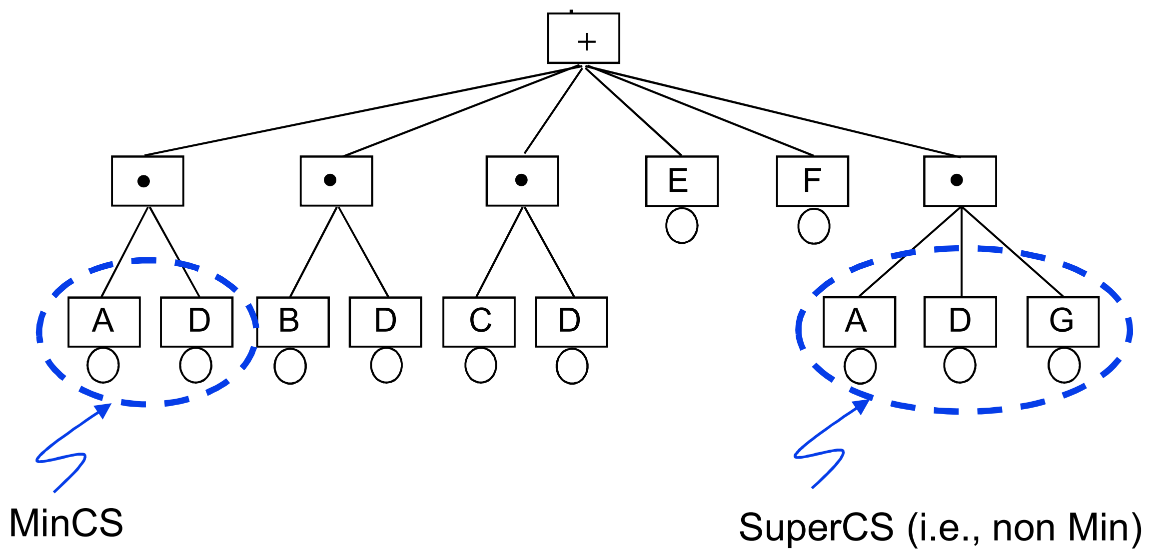
Cut Sets



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.

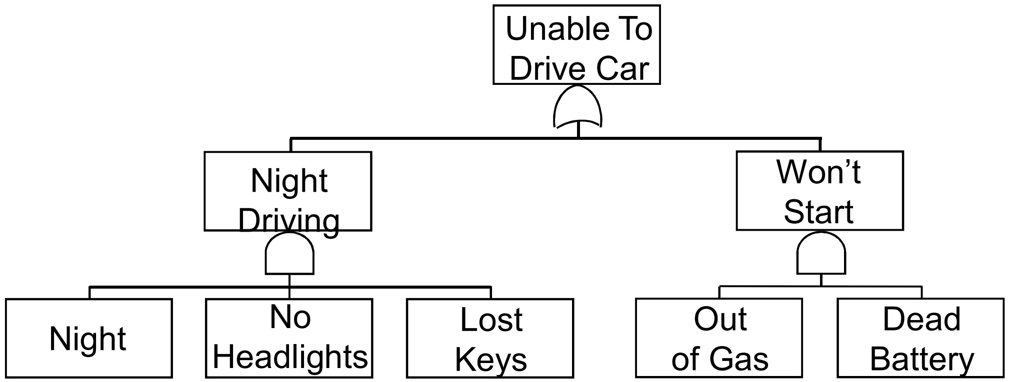
Min CS



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



If an item can be removed from CS and top still occurs then its not a Min CS.

CS1 - Night **&** No Headlights **&** Lost Keys

CS2 - Out of Gas **&** Dead Battery

← Invalid FT

(Not Min CS's)

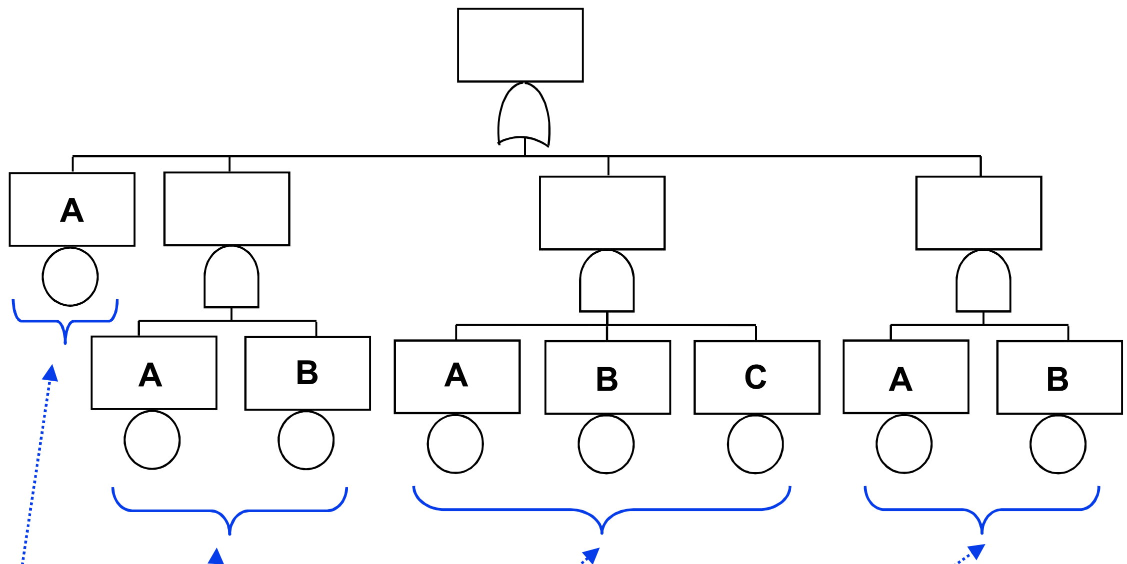
Should be:

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

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 ← **SCS**

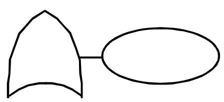
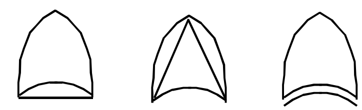
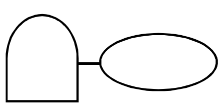
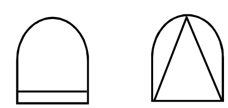
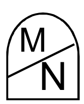
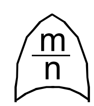
A,B,C ← **SCS**

A,B ← **DupCS, SCS**

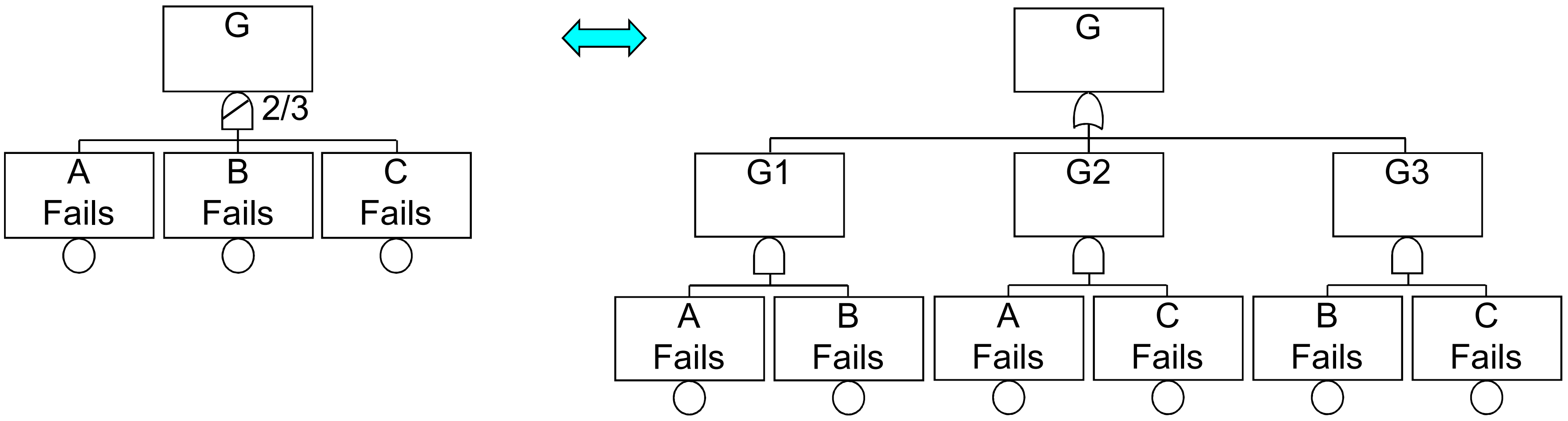
Min Cut Sets:

A

Alternate Gate Symbols

Symbol	Action	Description	Alternate Symbol
	Exclusive OR Gate	Only one of the inputs can occur, not both. Disjoint events.	
	Priority AND Gate	All inputs must occur, but in given order, from left to right.	
	M of N Gate	M of N combinations of inputs causes output to occur.	 Voting Gate

M/N Gate Example



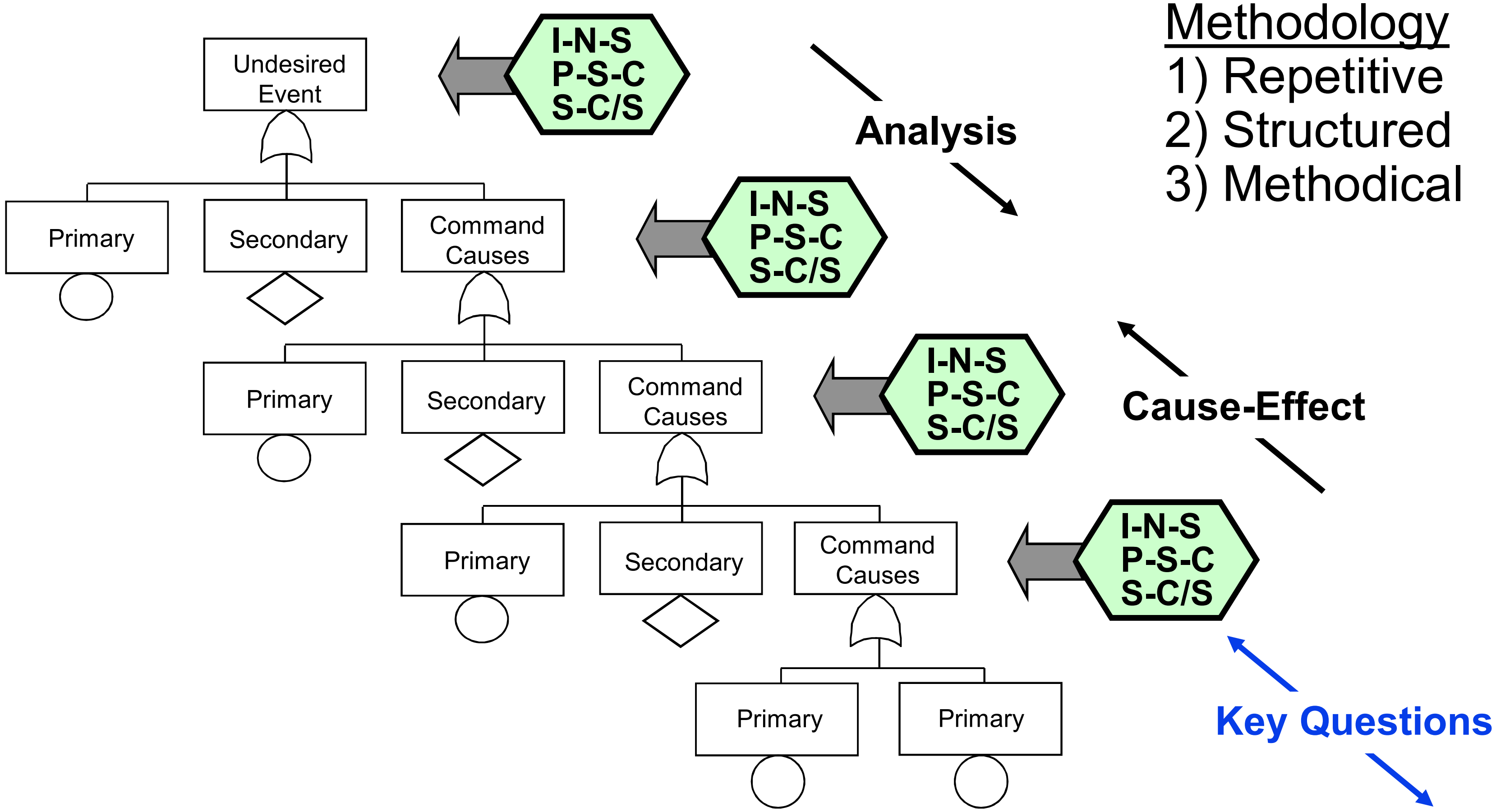
- M of N gate
- Also known As Voting gate

--- 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



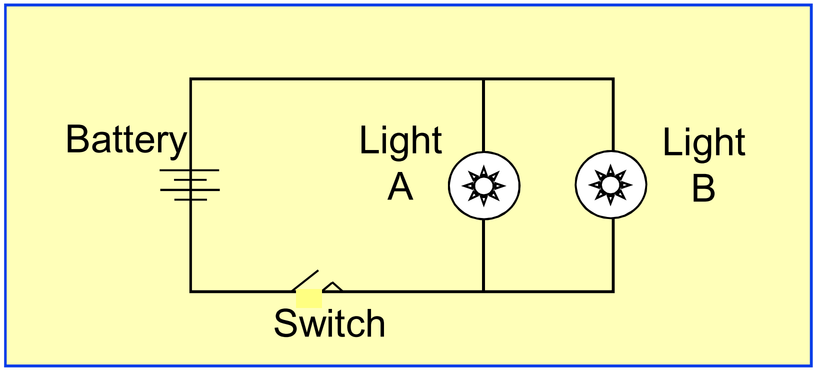
Methodology
1) Repetitive
2) Structured
3) Methodical

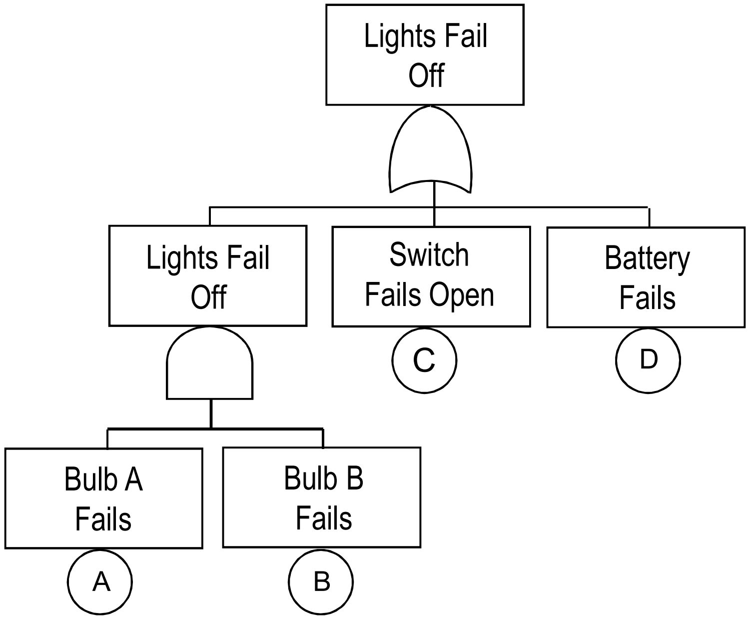
I-N-S=Immediate, Necessary, Sufficient
P-S-C=Primary, Secondary, Command
S-C/S=State of the Component or System

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

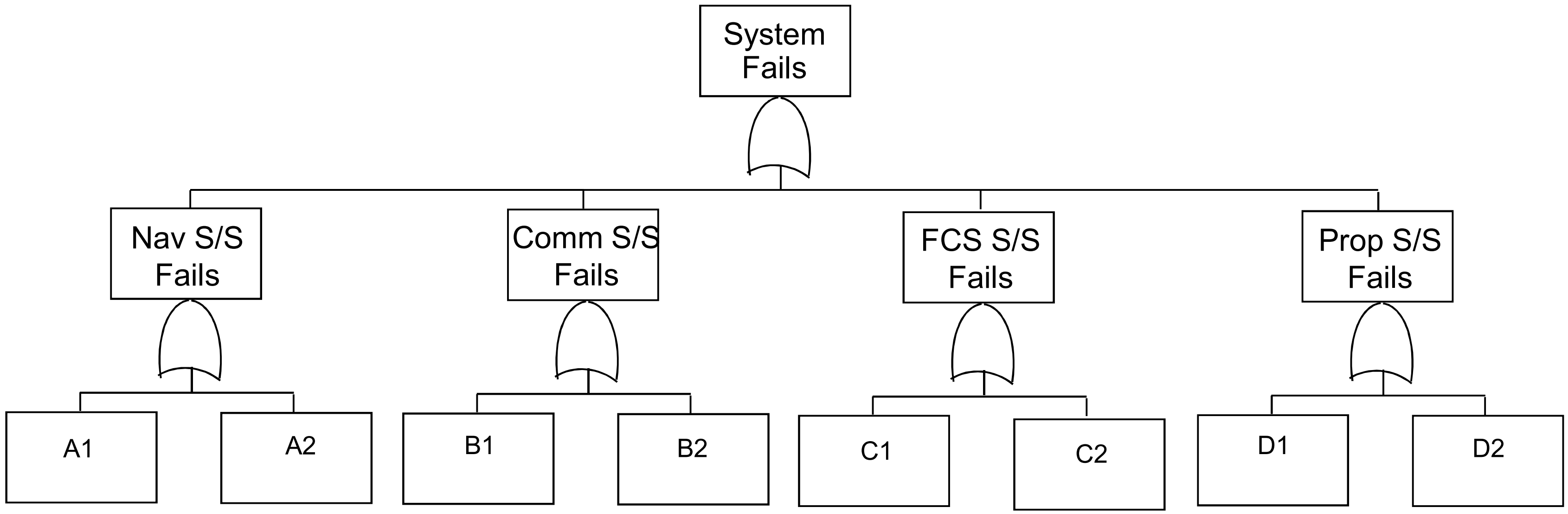




- Immediate breakdown by component
- Ignores immediate cause-effect relationships
- Tends to logically overlook things for large systems

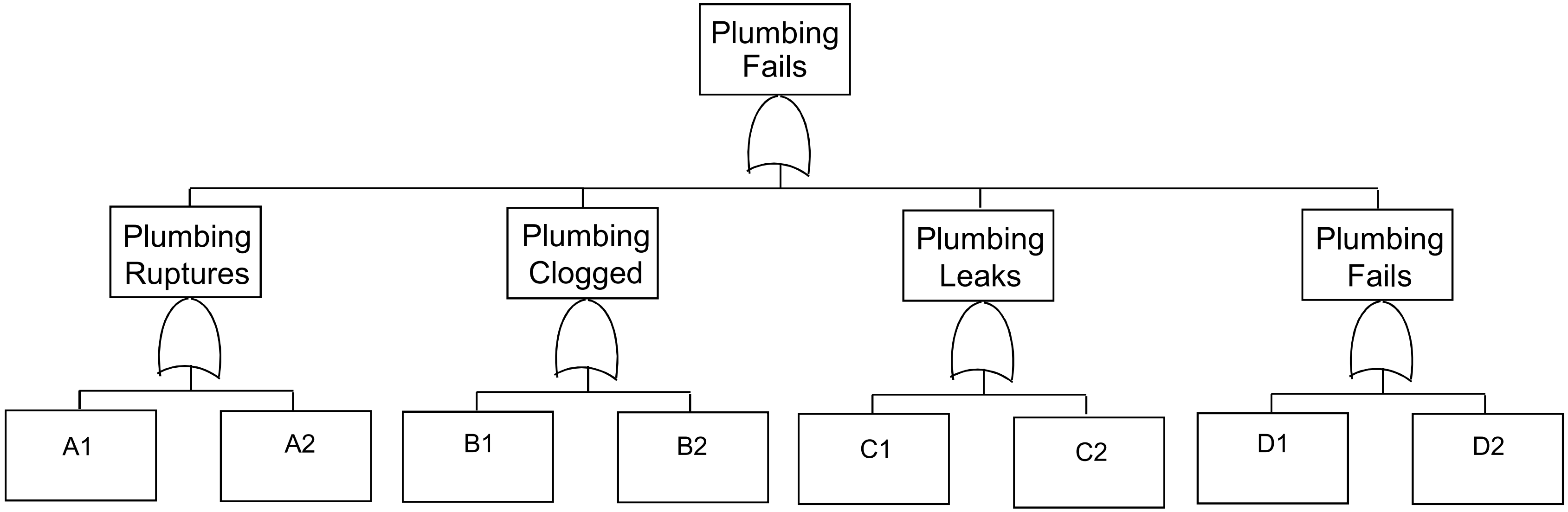
Subsystem Approach

- 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



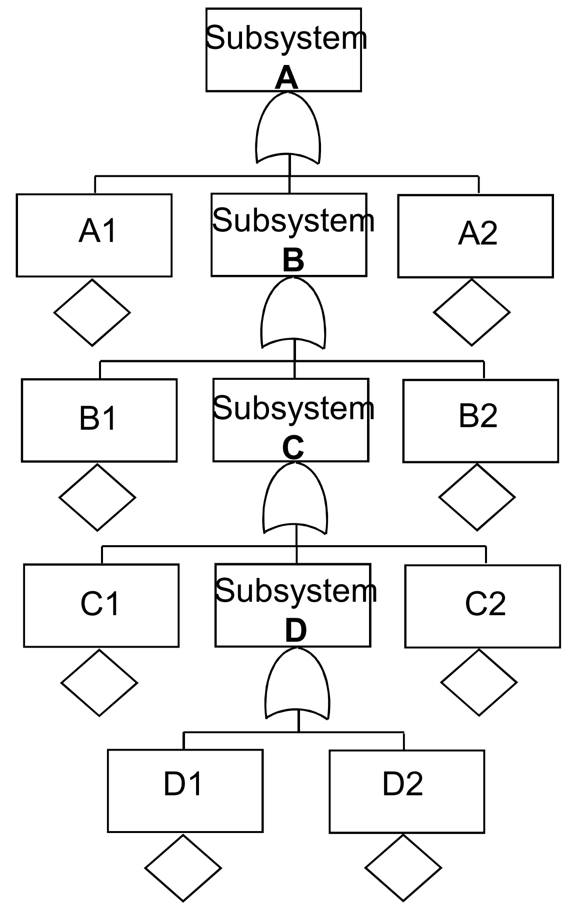
Scenario Approach

- 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



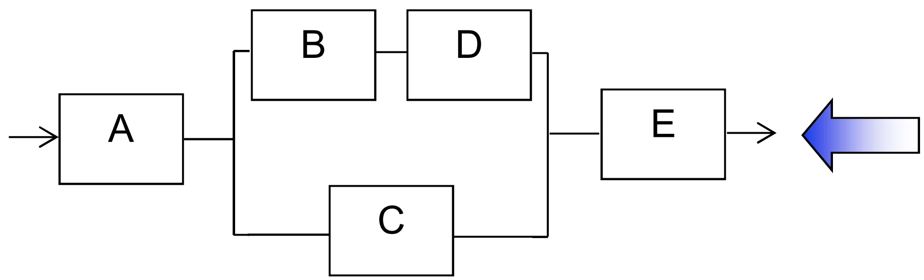
Functional Approach

- 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

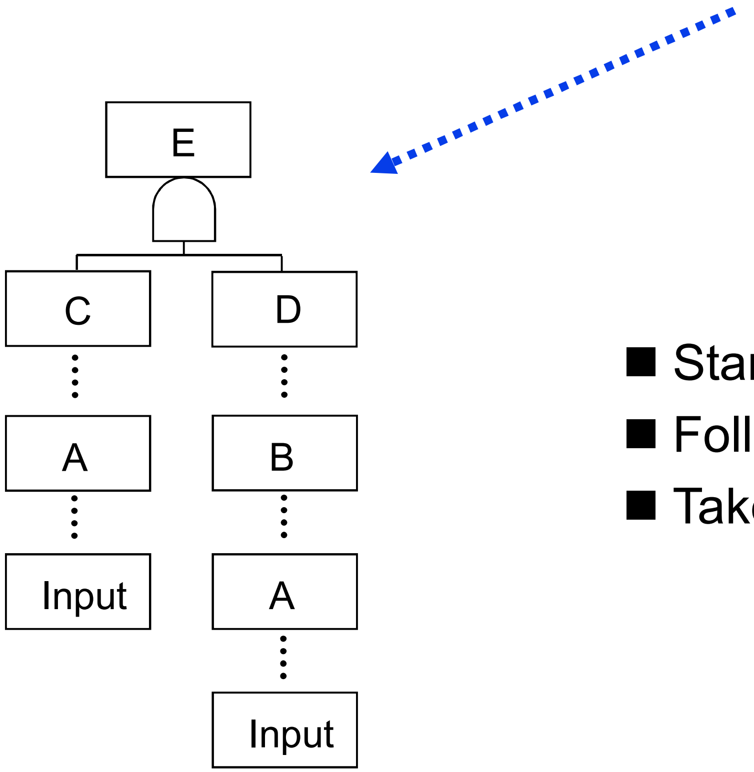


Recommended approach

Functional Approach

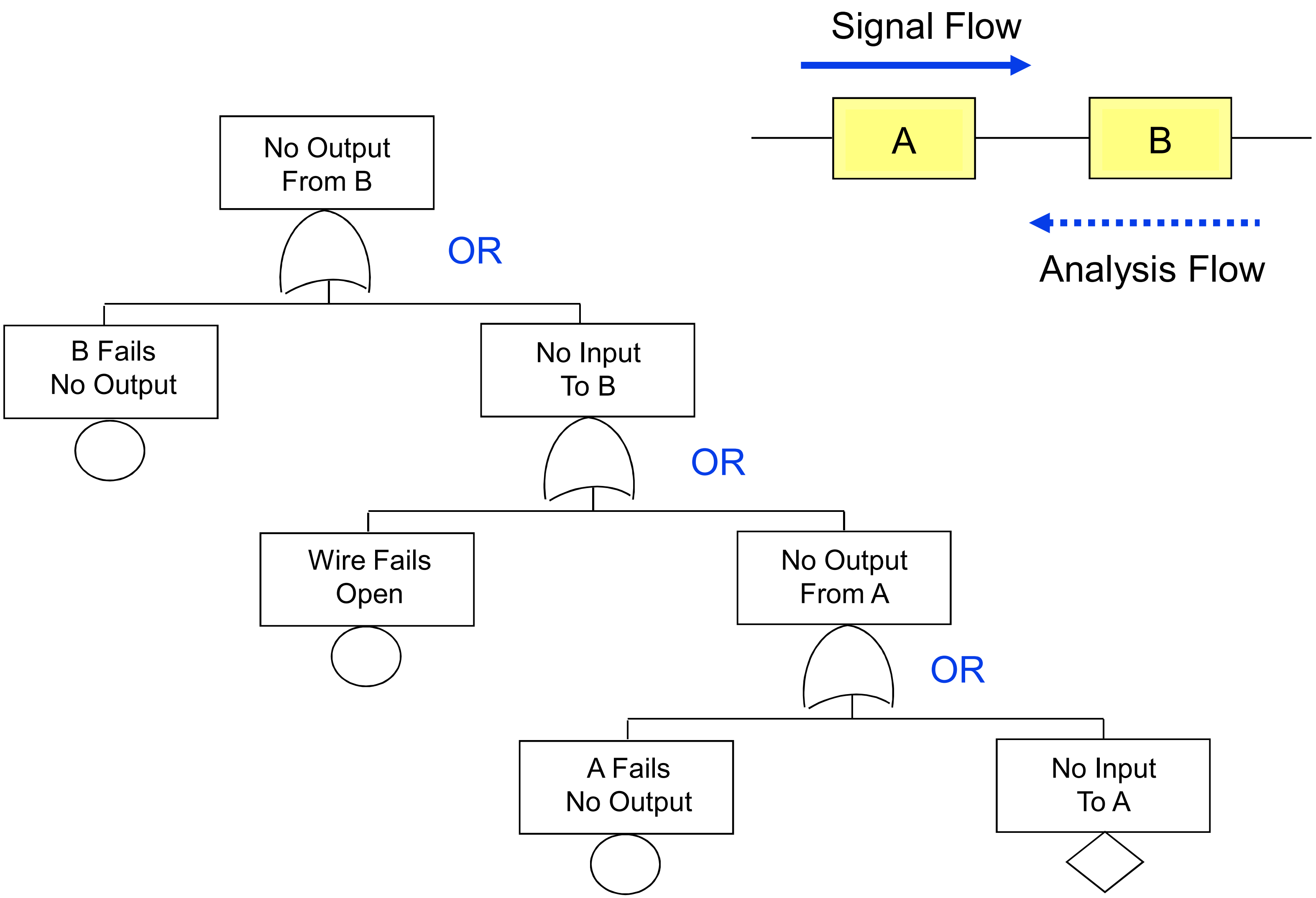


Follow the functional path

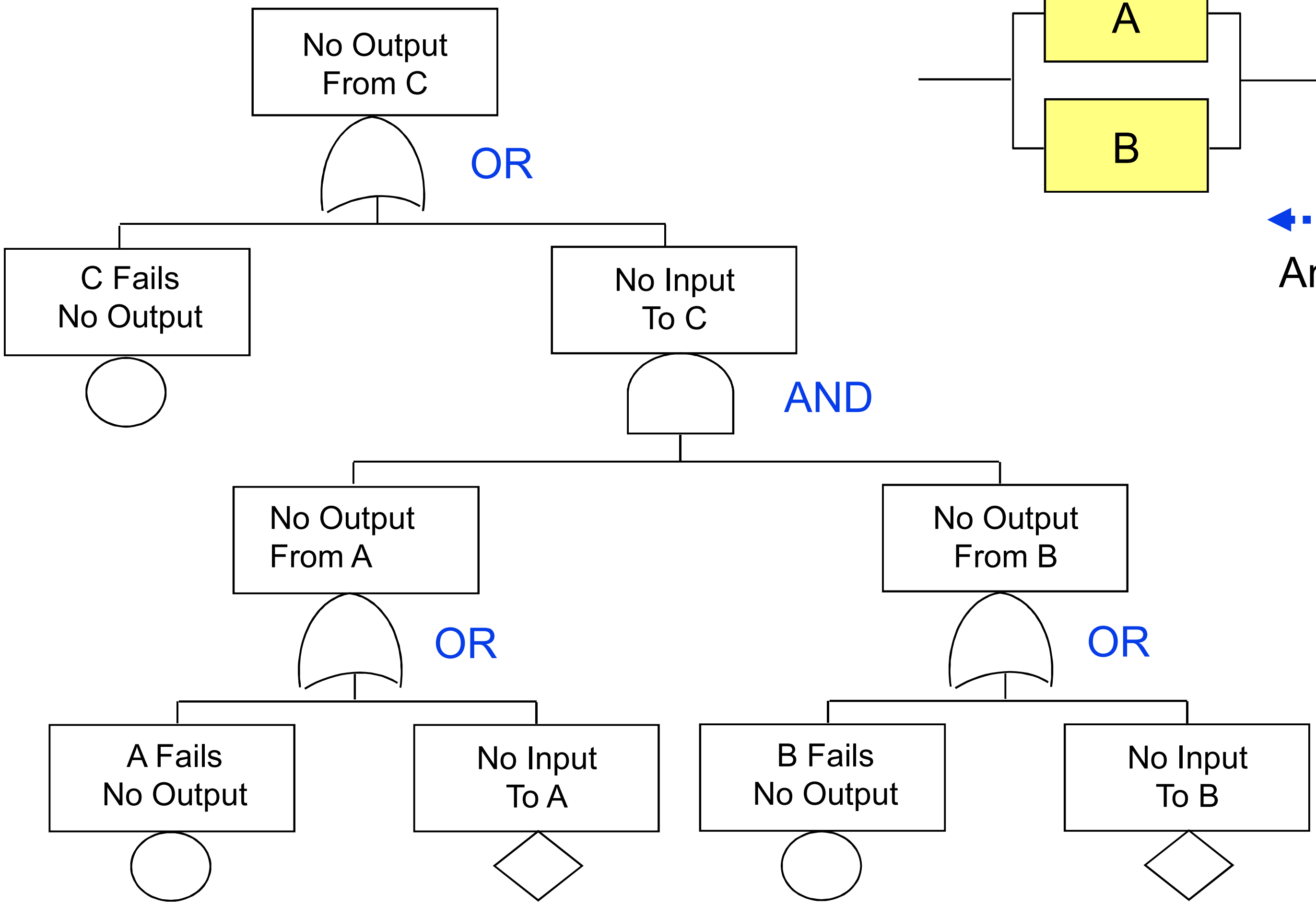


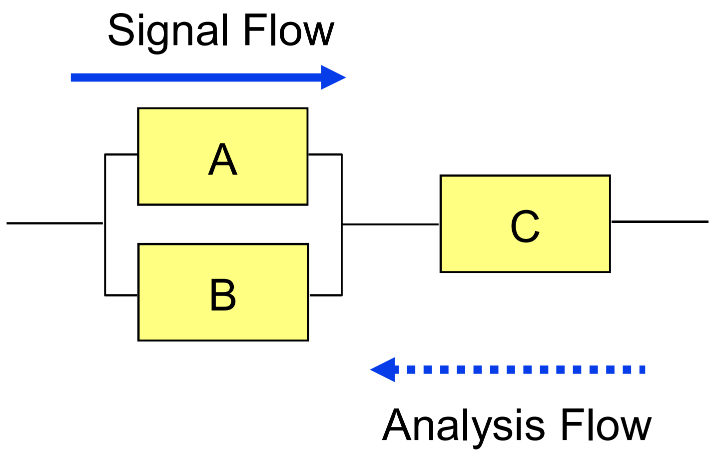
- Start at UE location (E in this example)
- Follow signal flow backwards
- Take each component one at a time

Series Example



Series-Parallel Example





FT Construction Methodology

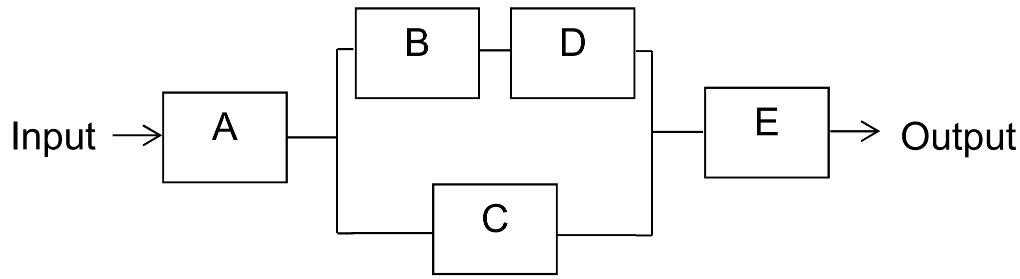
- Construction at each gate involves a 3 step question process:
  - Step 1 – Immediate, Necessary and Sufficient (I-N-S) ?
  - Step 2 – Primary, Secondary and Command (P-S-C) ?
  - Step 3 – State of the Component or System (S-C/S) ?

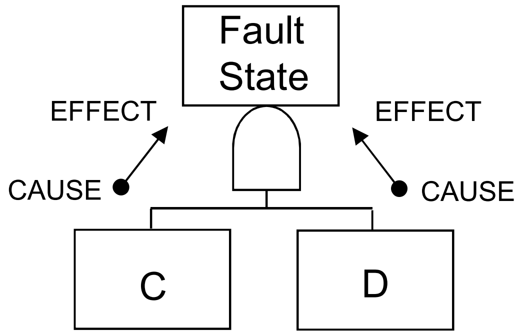
These are the 3 key questions in FTA construction

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.

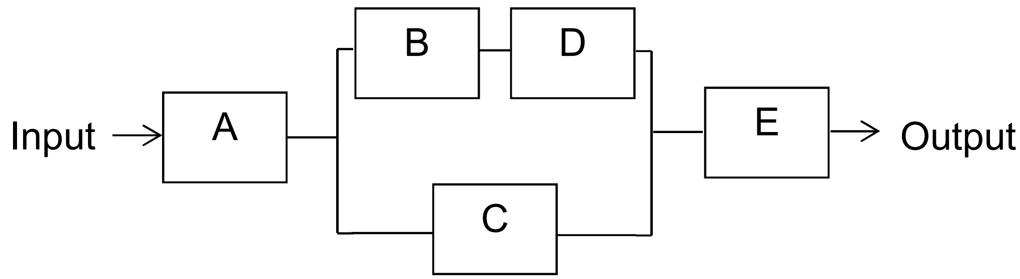
} To cause Fault of E

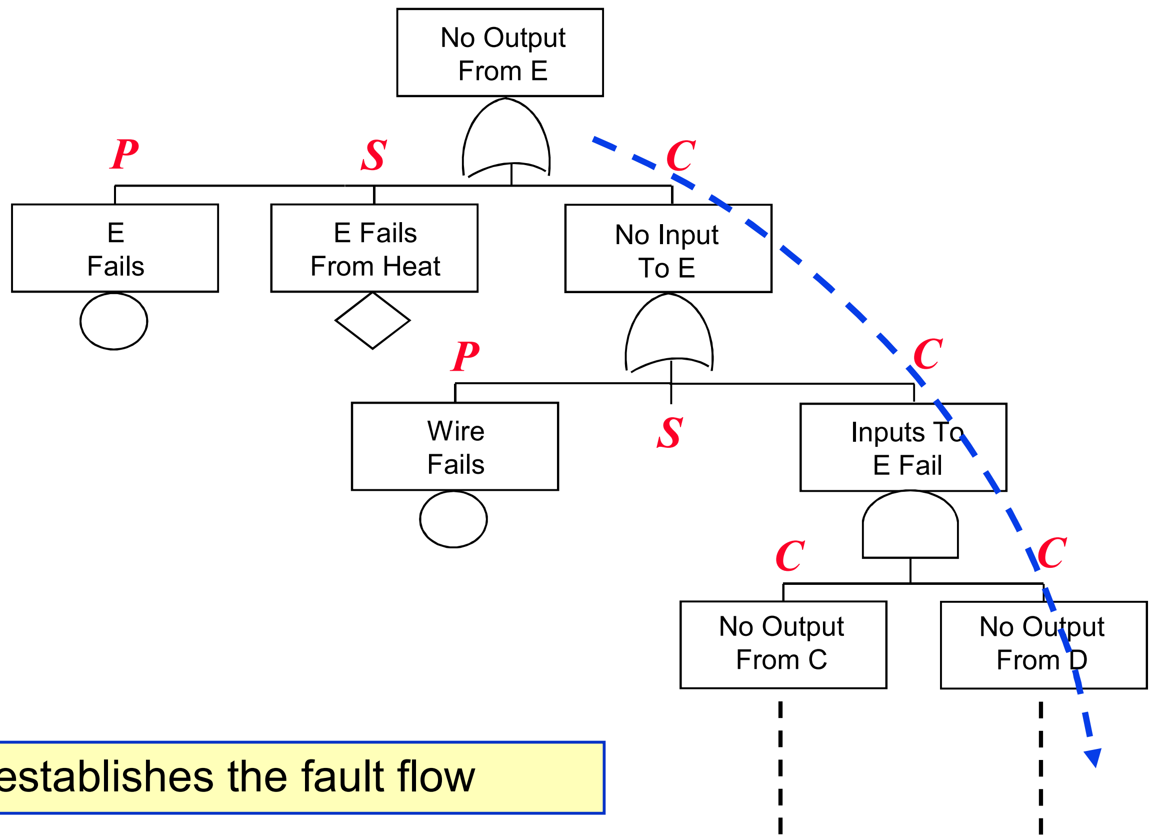
Step 2

- 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

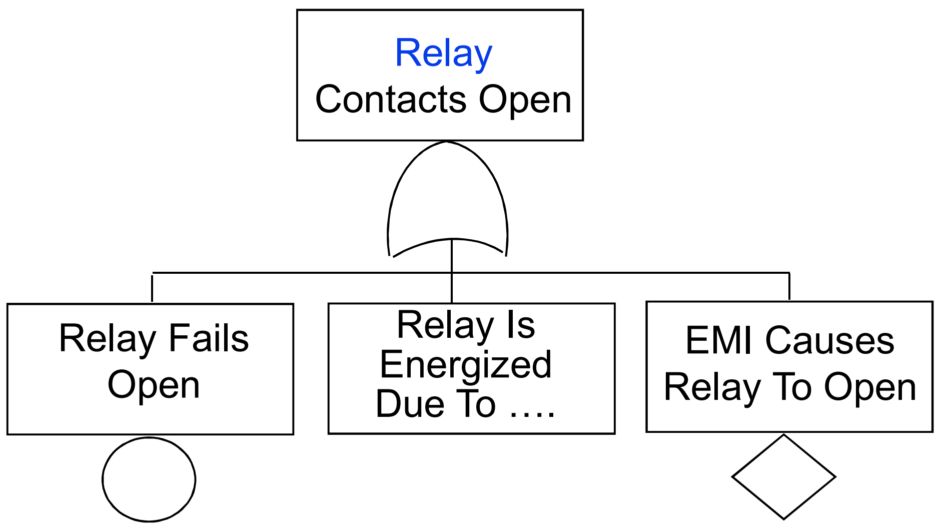
The Command path establishes the fault flow

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

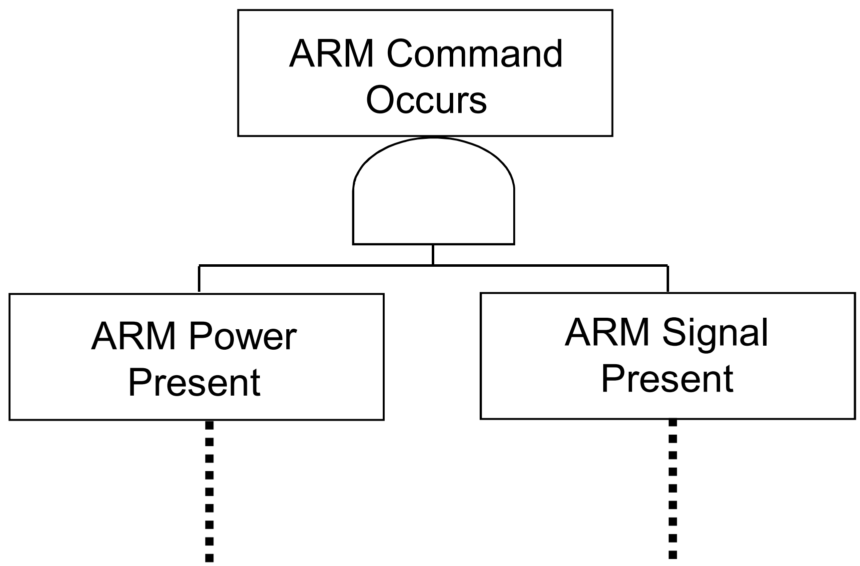
Step 3 (continued)

- 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

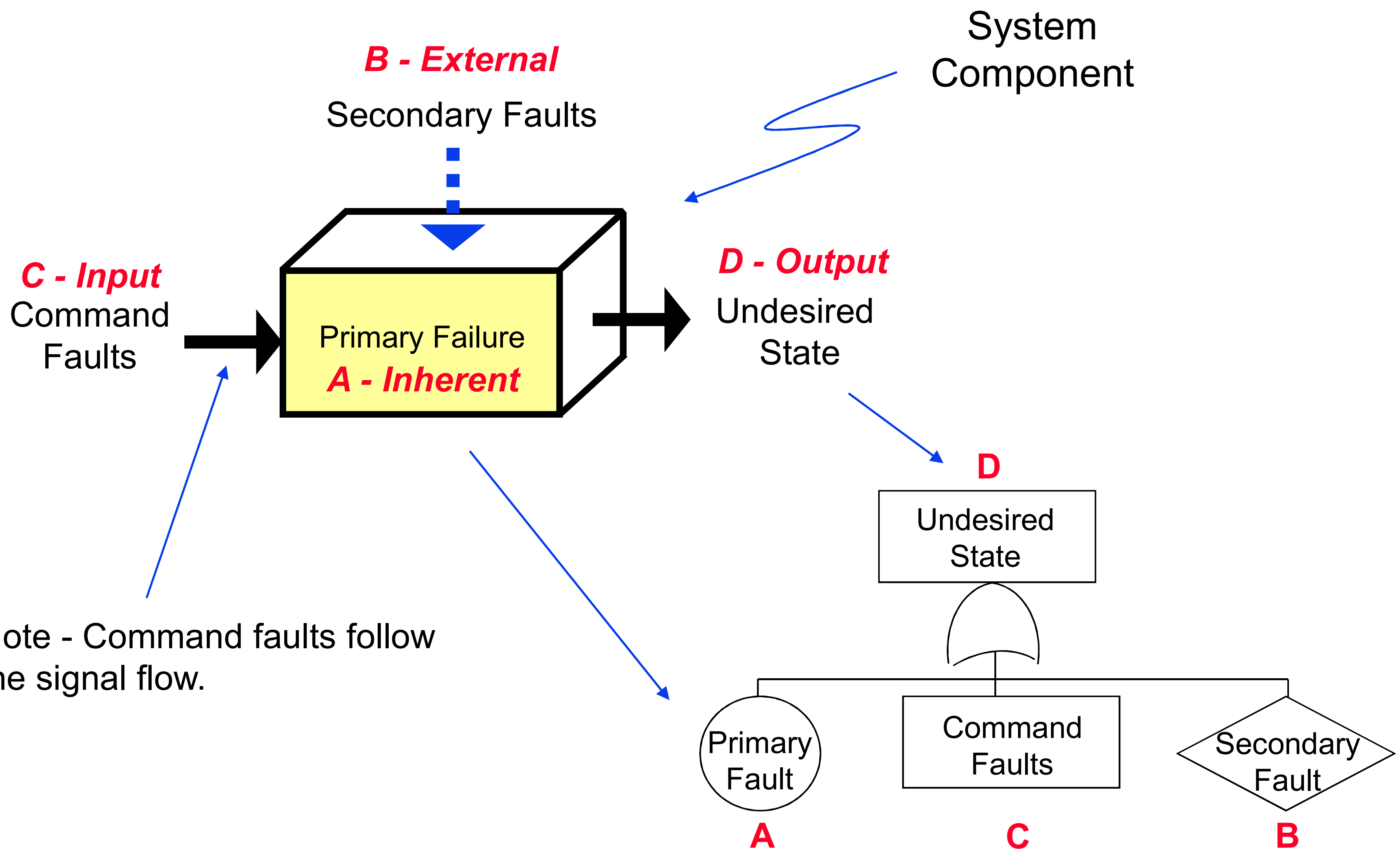


Step 3 (continued)

- 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

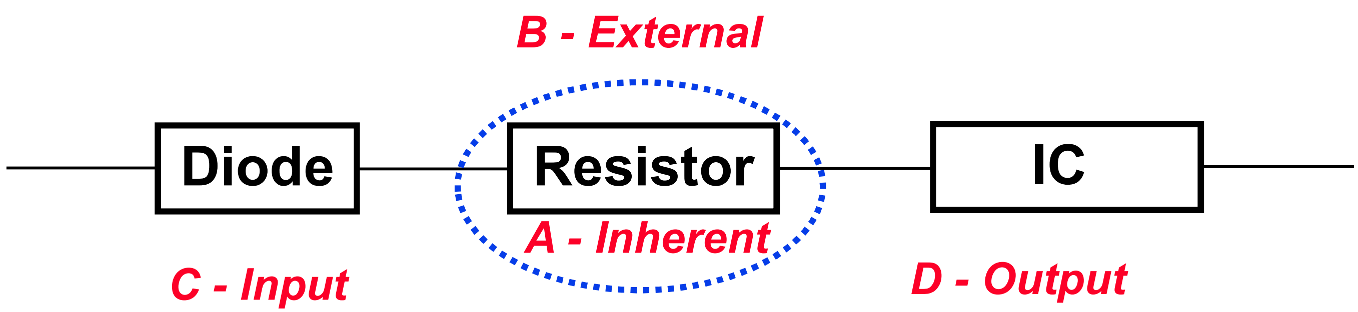


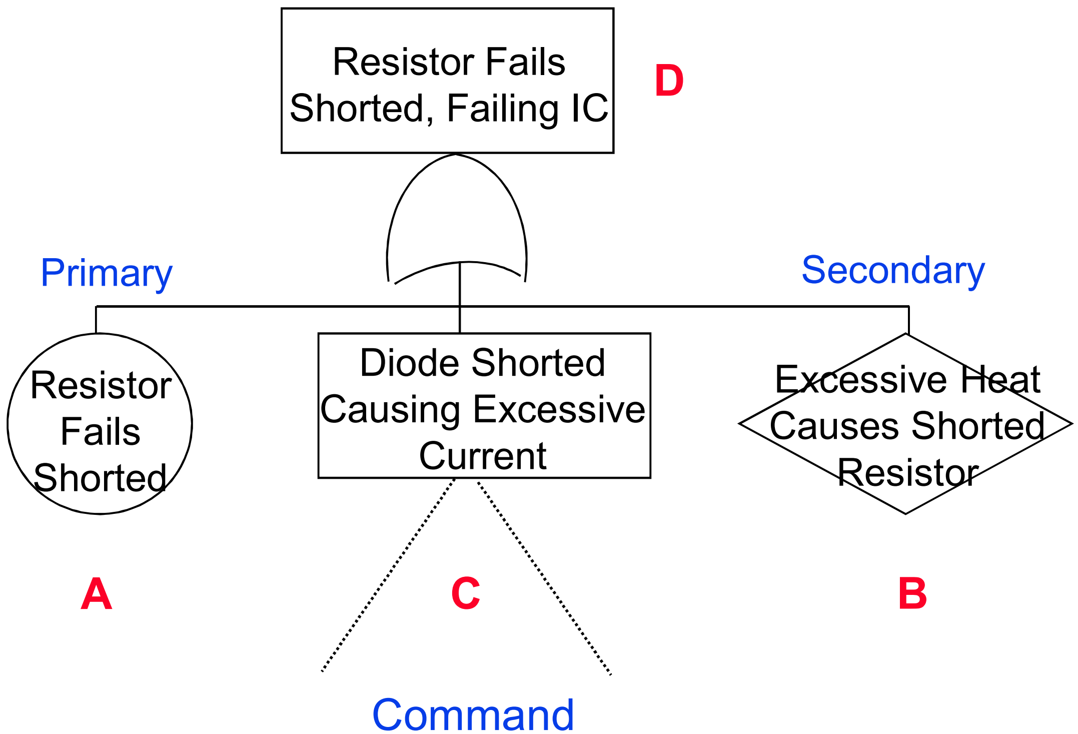
P-S-C Relationship to FT Structure



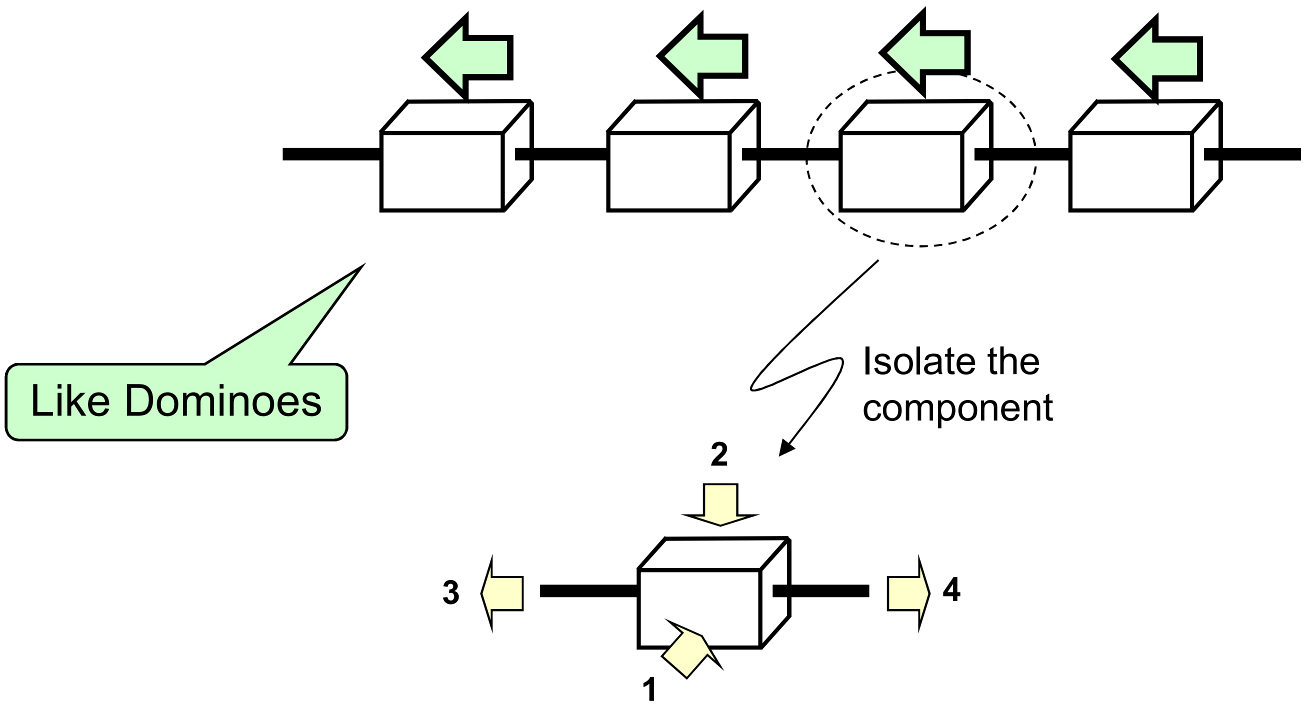
Note - Command faults follow the signal flow.

P-S-C Example


B - External
Diode
Resistor
IC
A - Inherent
C - Input
D - Output


Resistor Fails Shorted, Failing IC
D
Primary
Secondary
Resistor Fails Shorted
Diode Shorted Causing Excessive Current
Excessive Heat Causes Shorted Resistor
A
C
B
Command

Isolate and Analyze



Analysis Views:

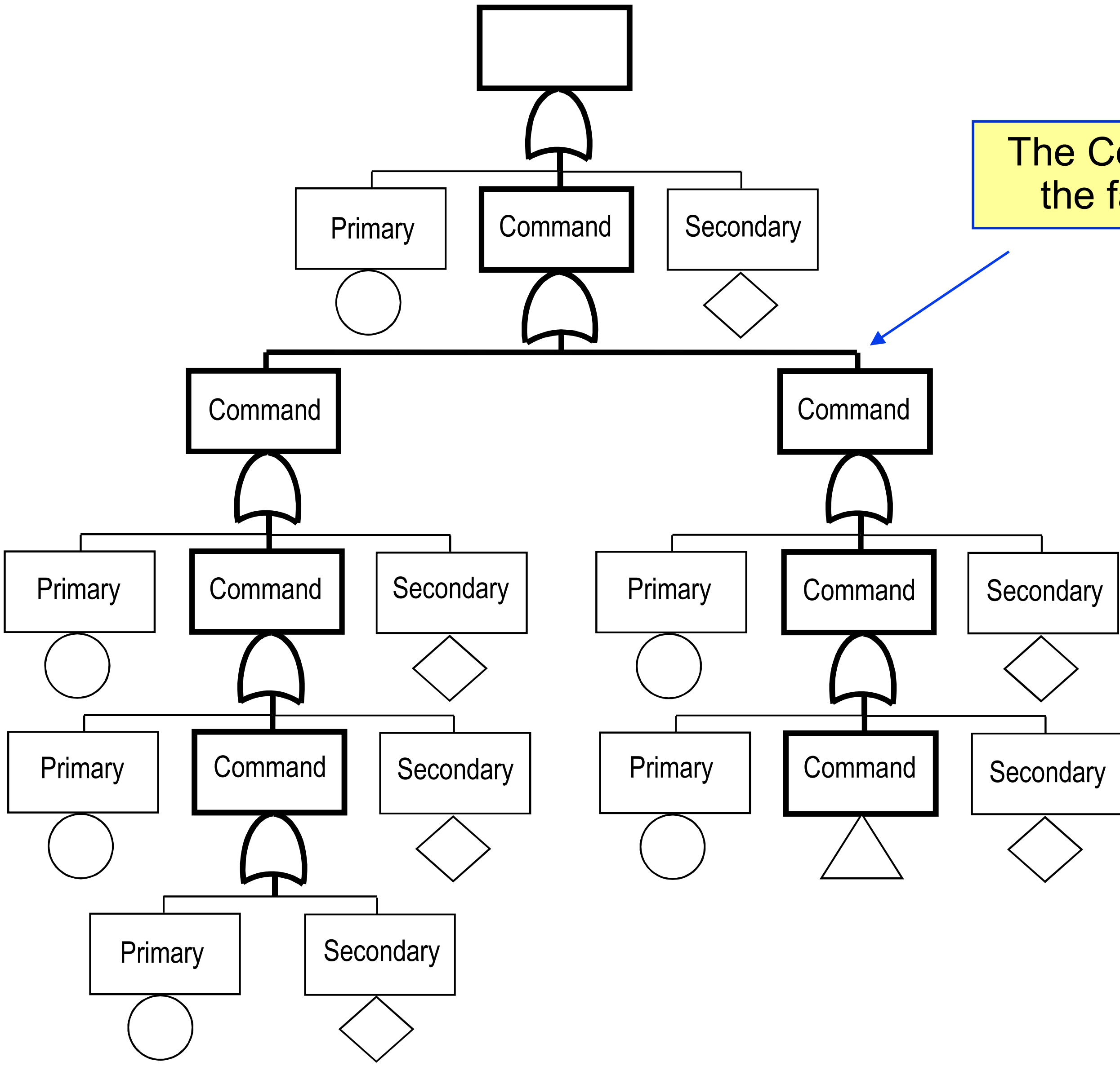
1) Primary - look inward

2) Secondary - look outward for incoming environmental concerns

3) Command - look backward at incoming signals

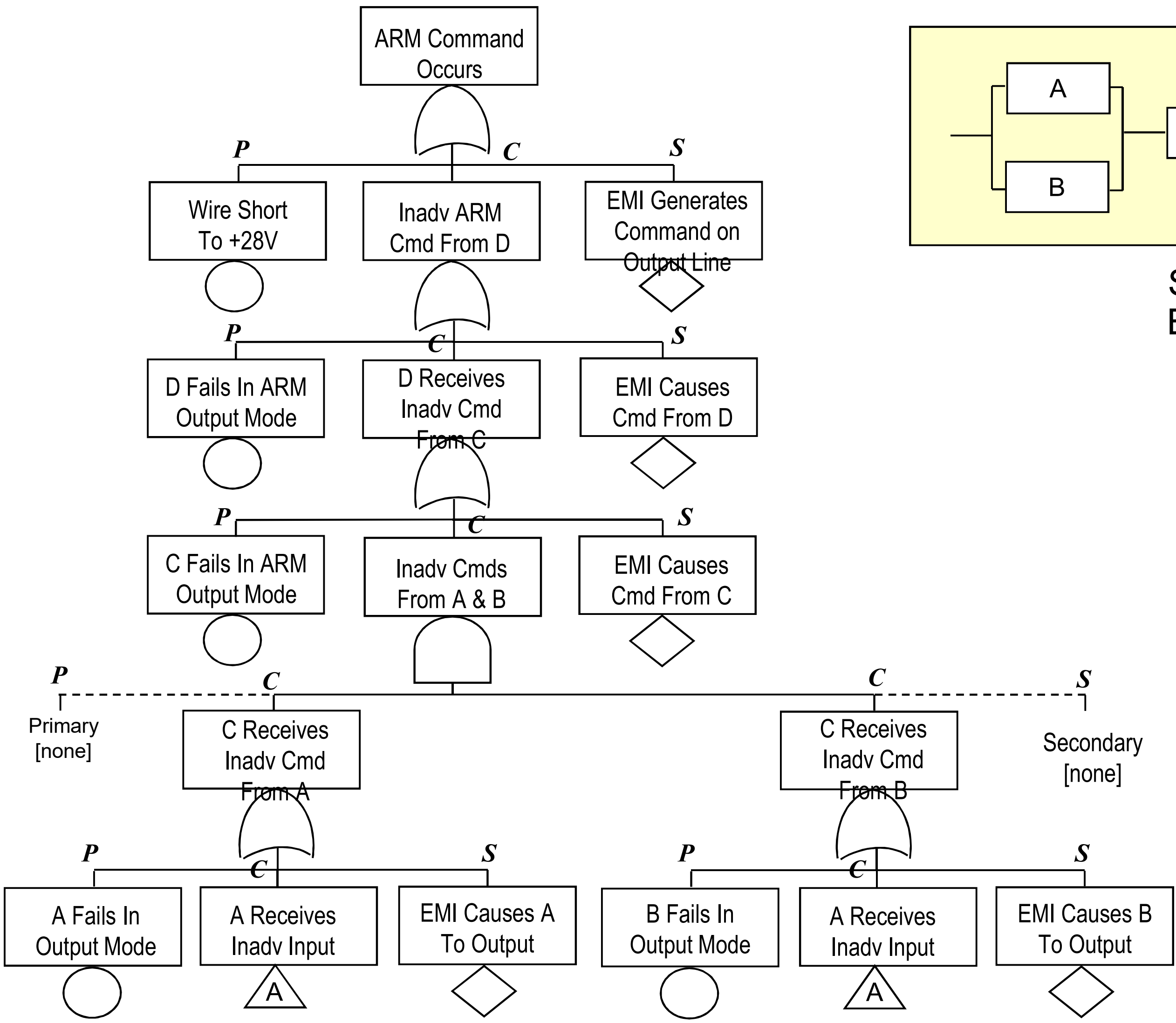
4) Output - look forward at possible undesired states that can be output

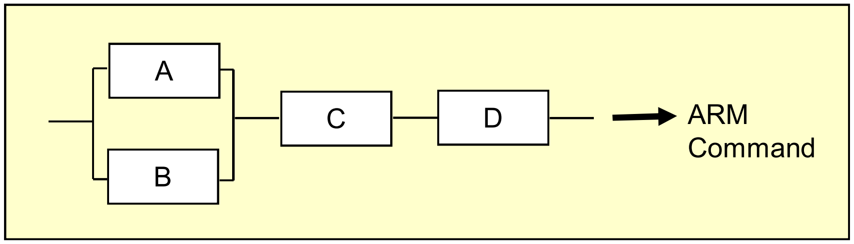
Example of Command Path



The Command path establishes the fault flow through the FT

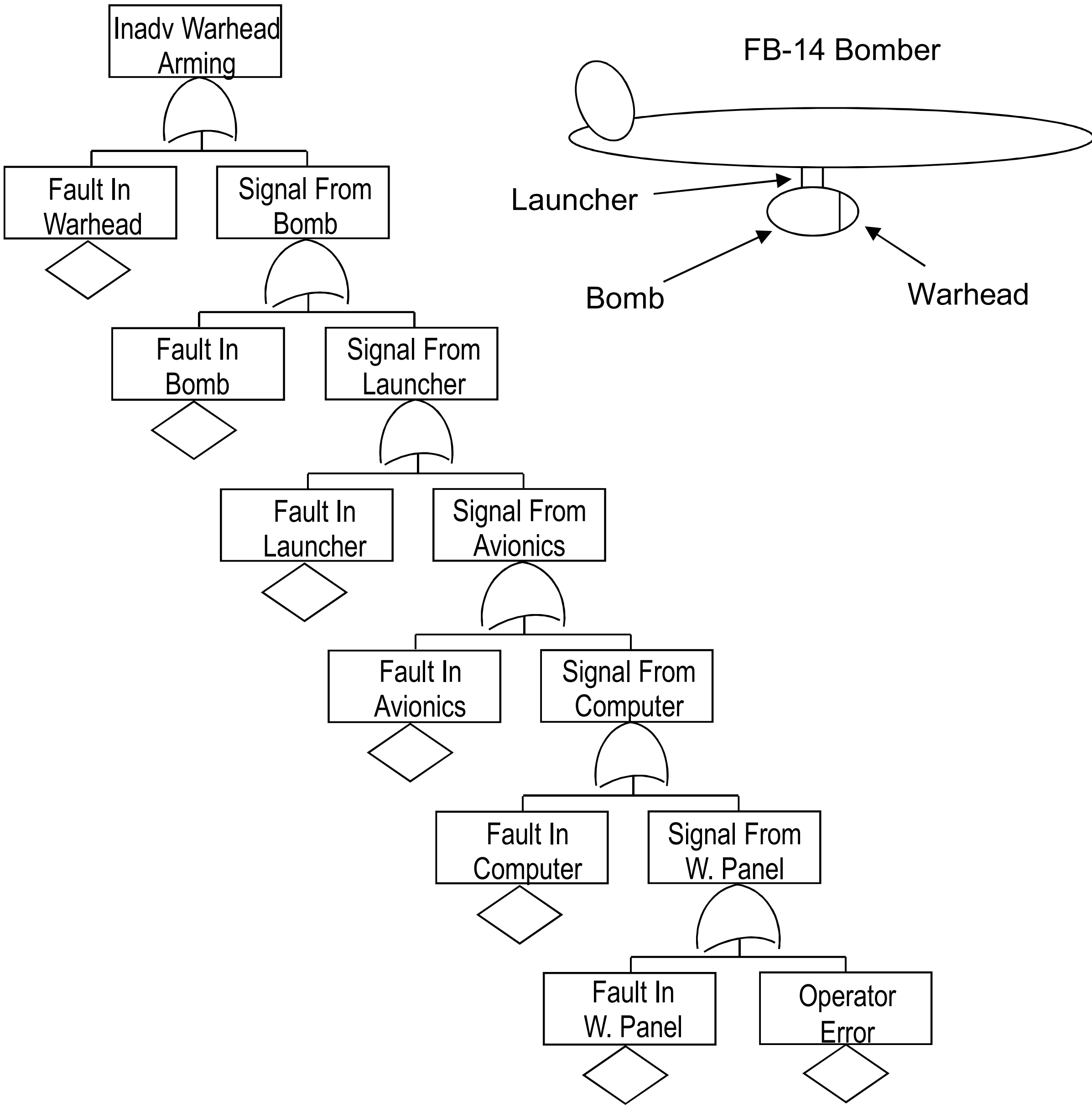
Example

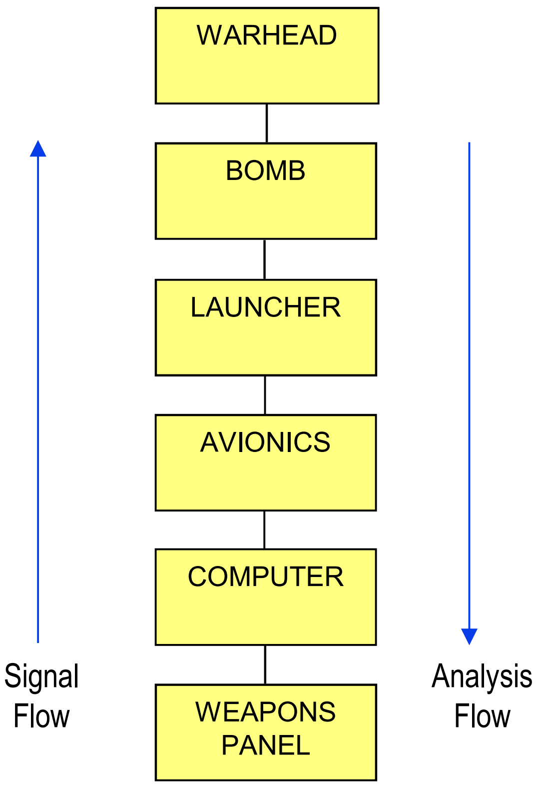




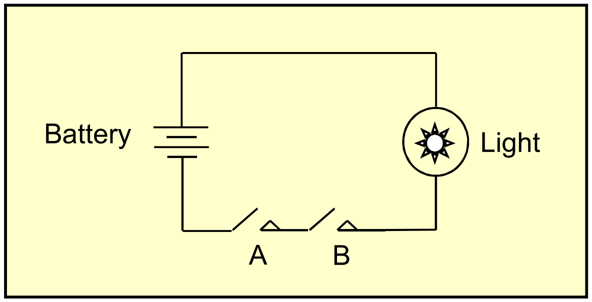
System design -- both A and B are necessary to cause C.

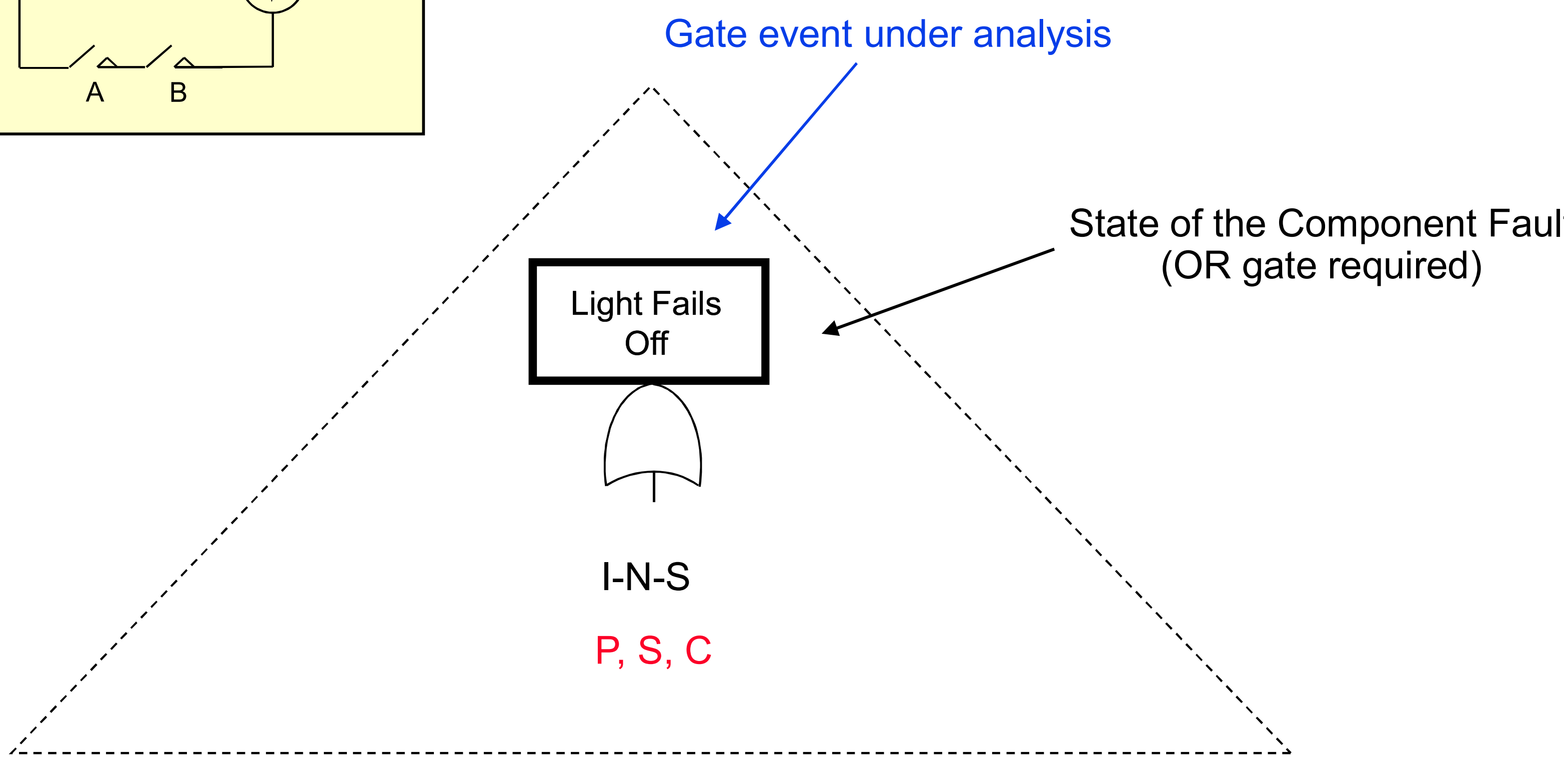
Example Fault Tree





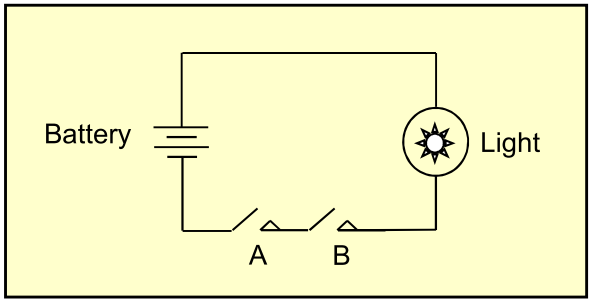
Construction Example

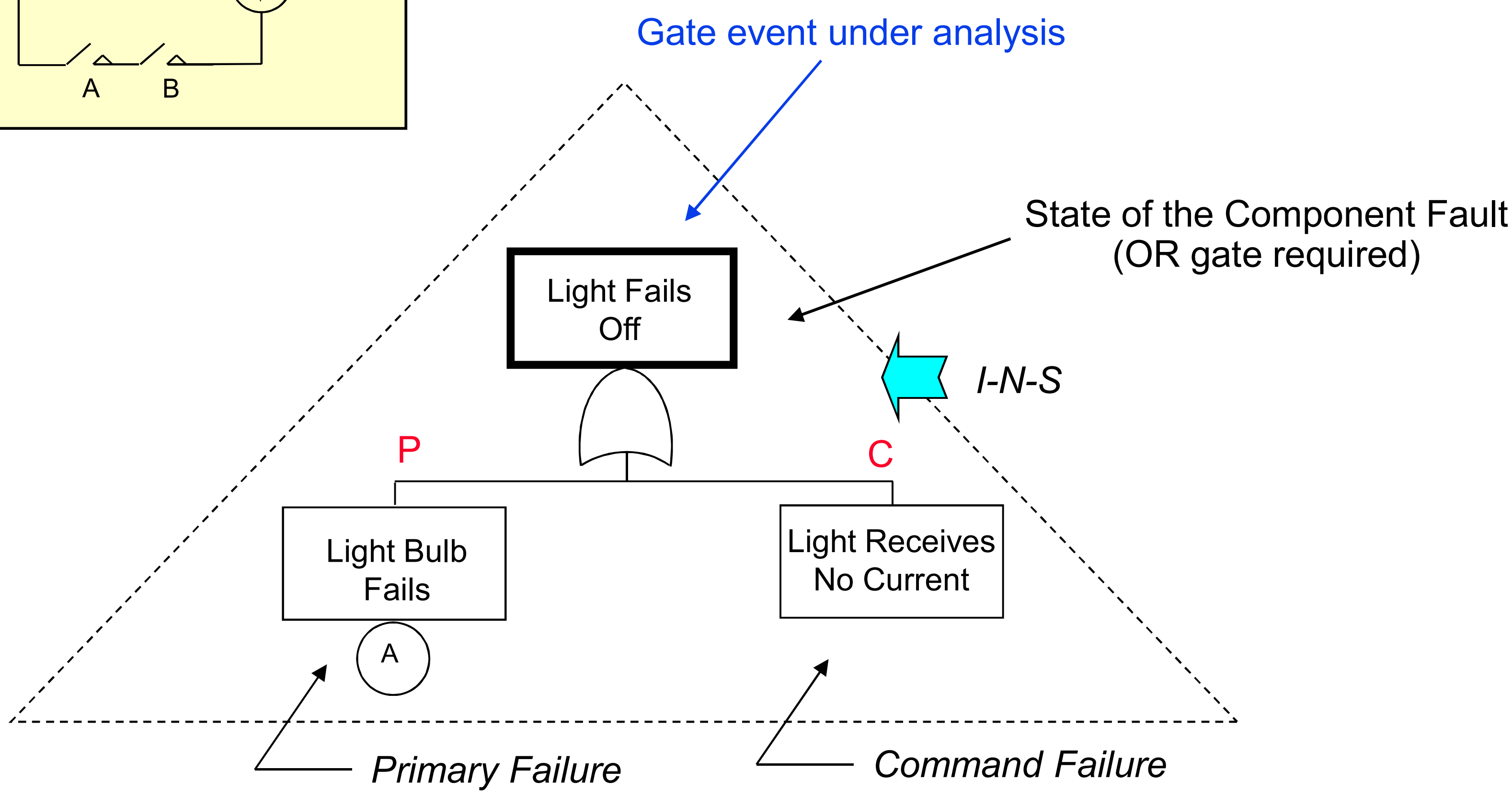




P – primary failure
S – secondary failure
C – command fault

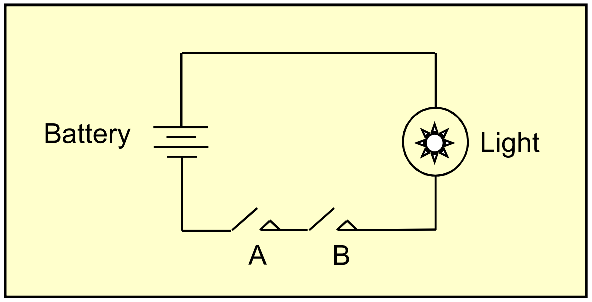
Construction Example

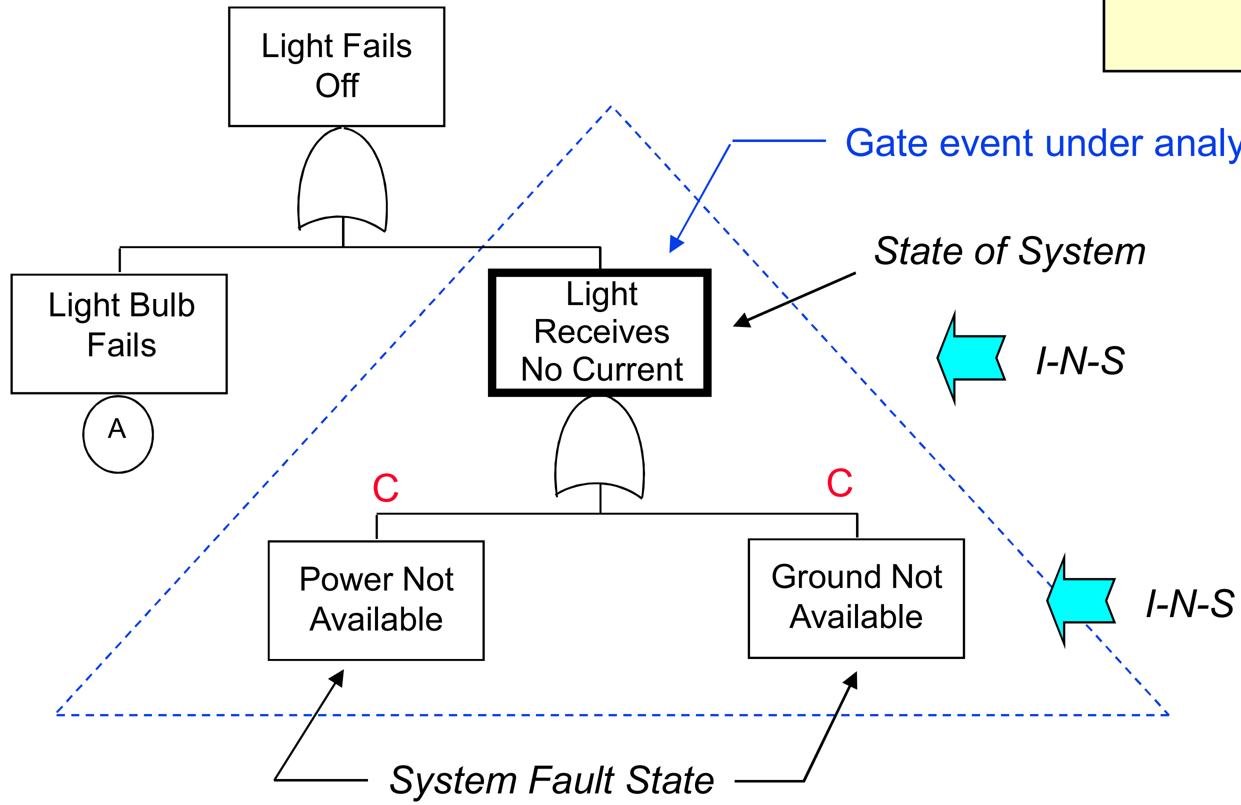




Note – This uses P-S-C, I-N-S and S-C/S

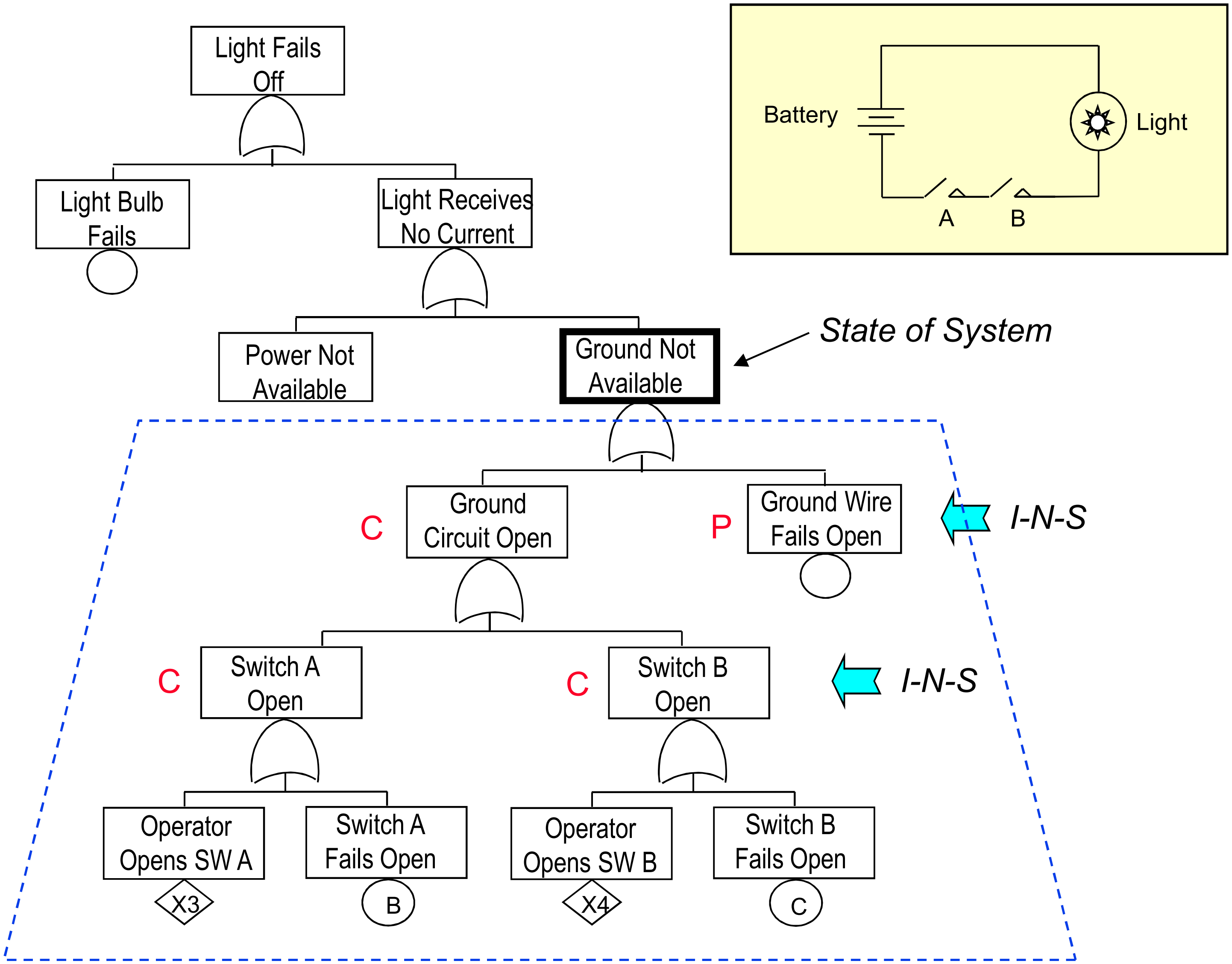
Construction Example (continued)



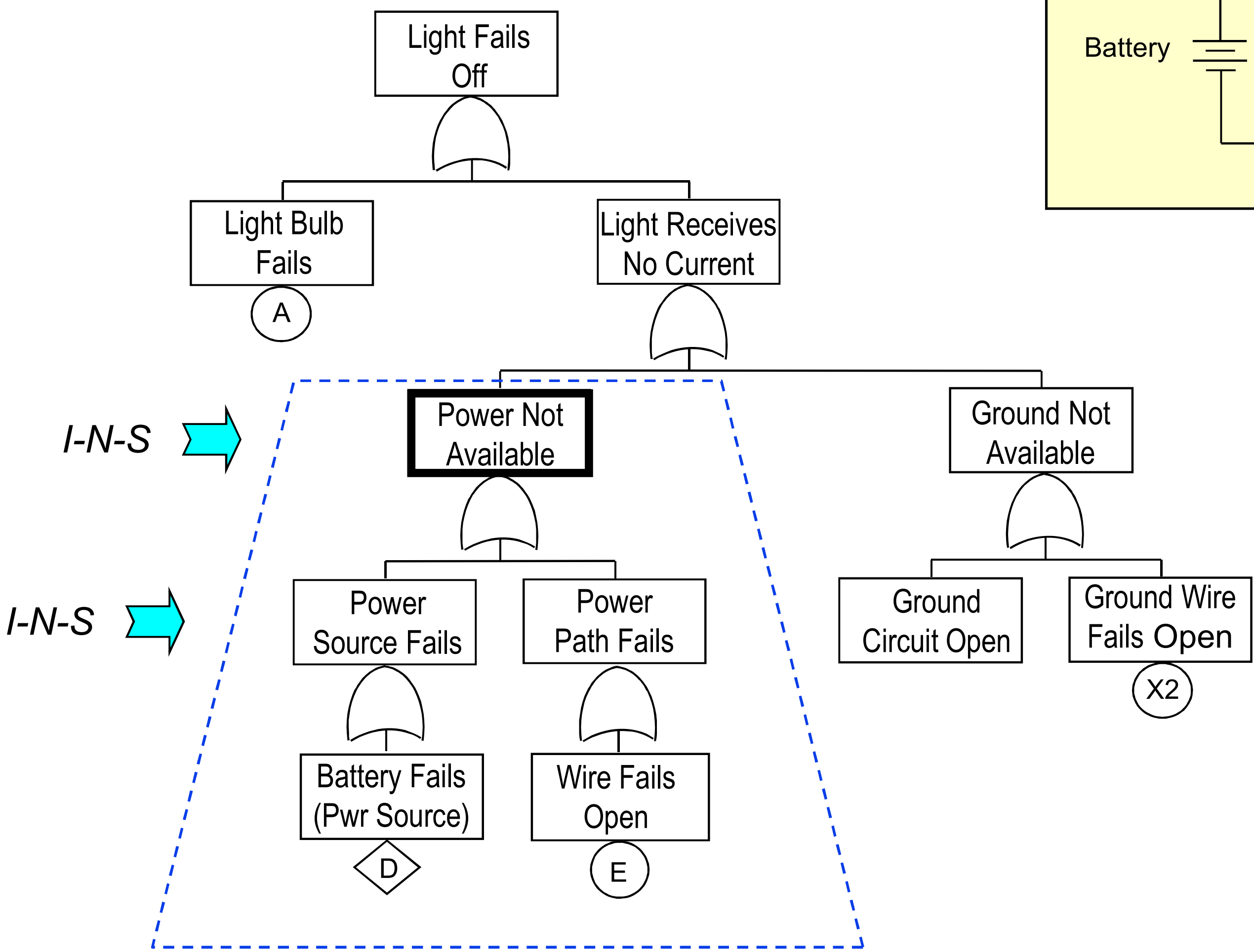


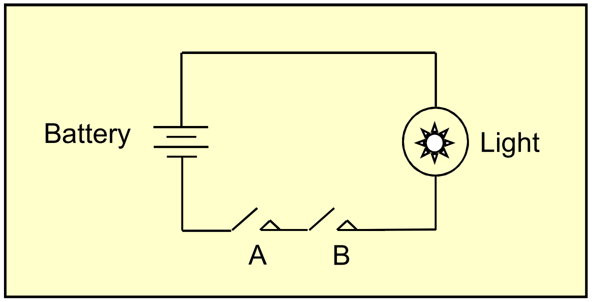
C – command fault

Construction Example (continued)

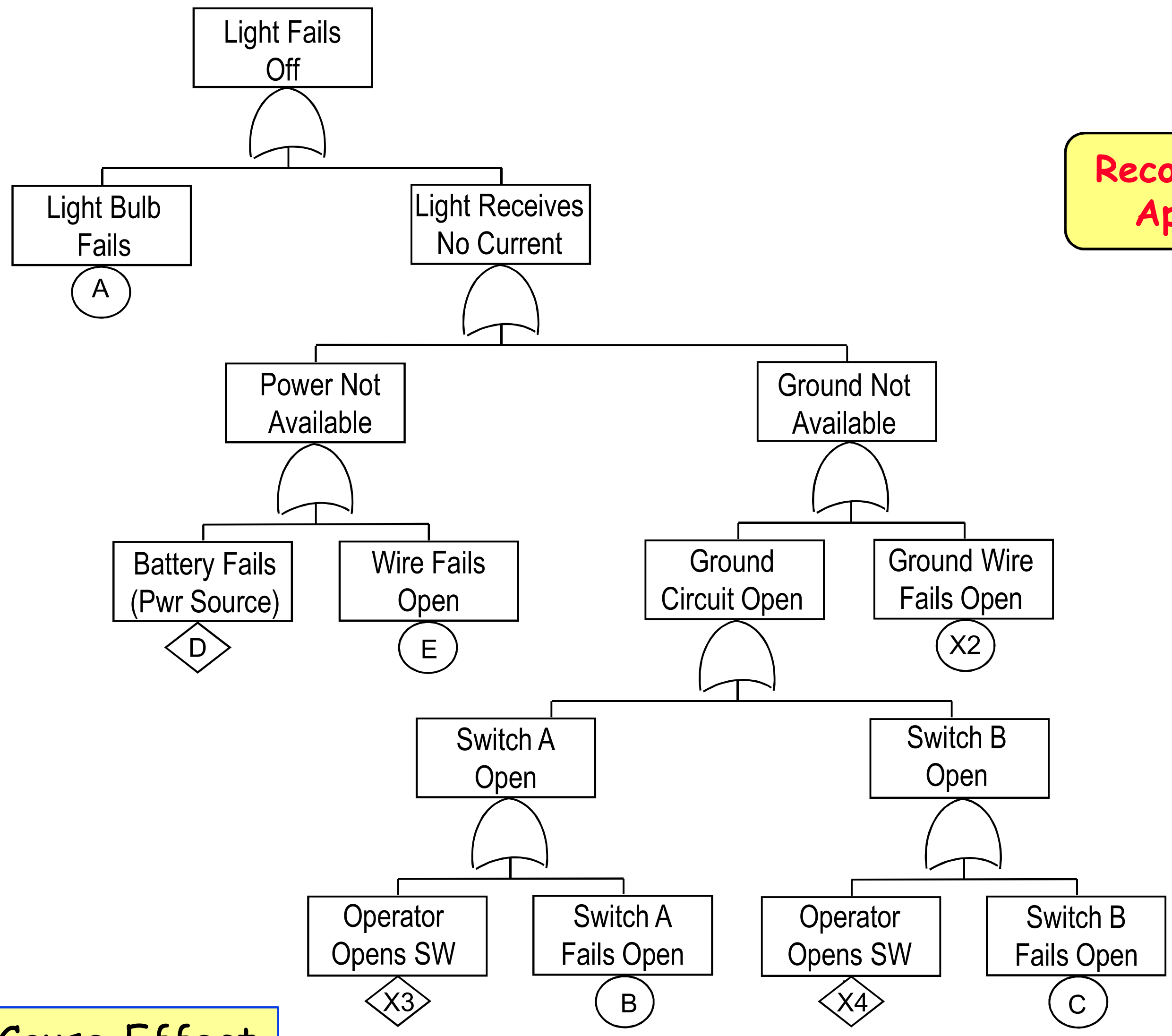


Construction Example (continued)





FT Process – Functional Flow



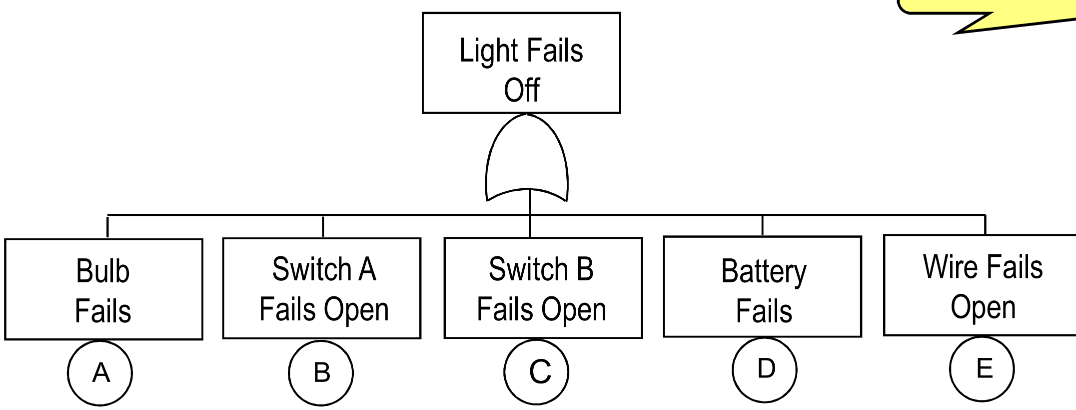
Recommended Approach

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



Note that Cause-Effect relationship is **not** visible

--- FT Construction Rules ---

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

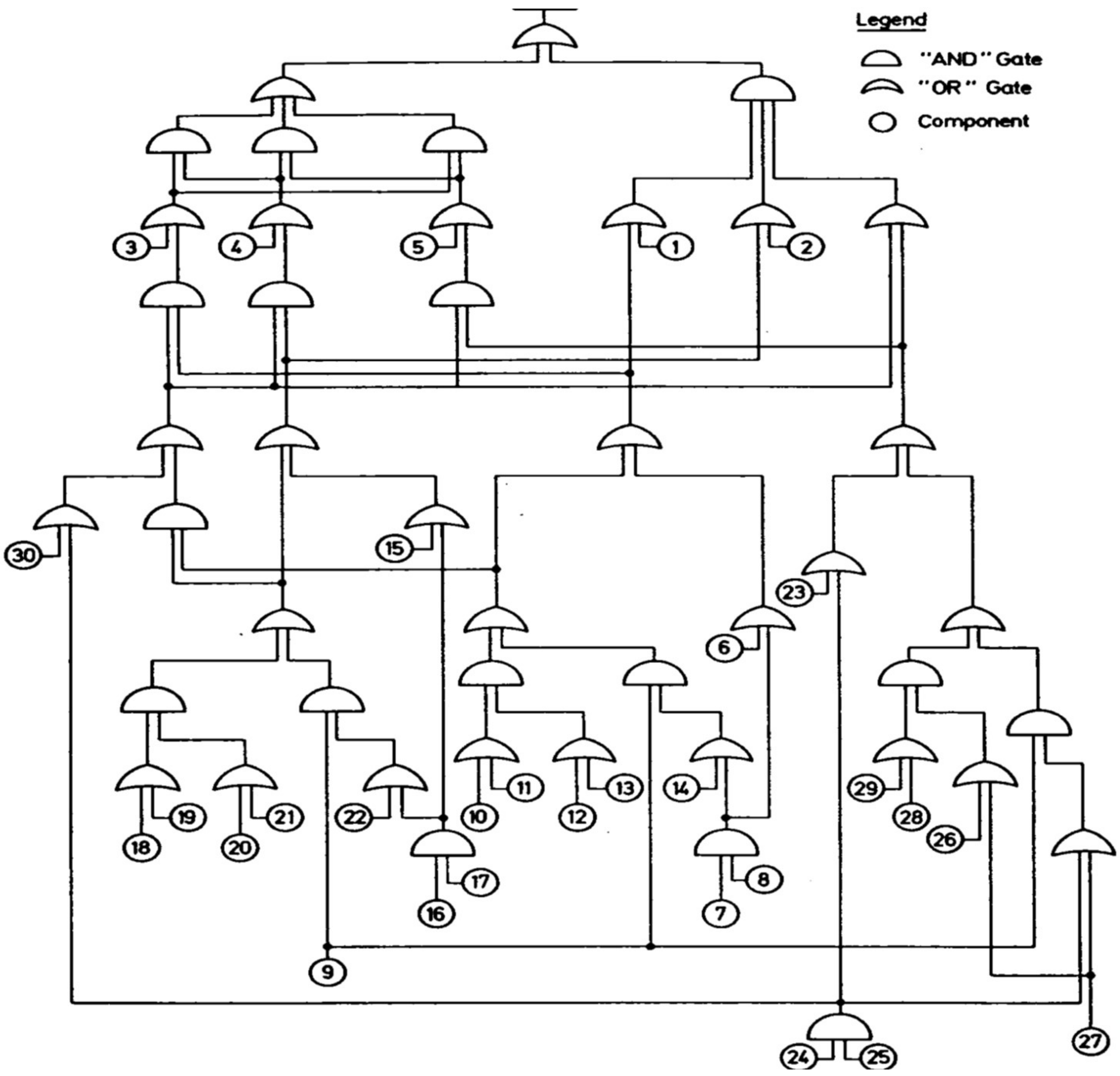
FT size is important and has many implications

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

9 - Think in Terms of Failure Space

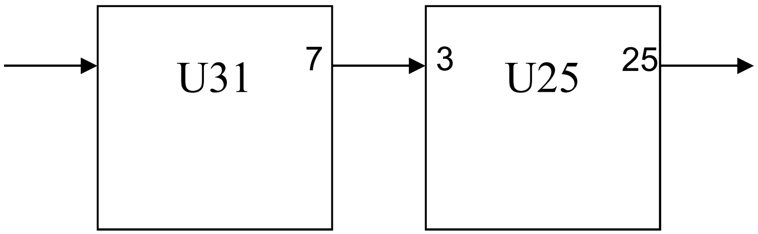
- Remember, it’s a ”fault” tree, not a “success” tree
  - Analysis of failures, faults, errors and bad designs
- No magic
  - Do not draw the fault tree assuming the system can be saved by a miraculous failure
  - This is normally referred to as the “No Magic Rule”
- No operator *saves*
  - When constructing 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

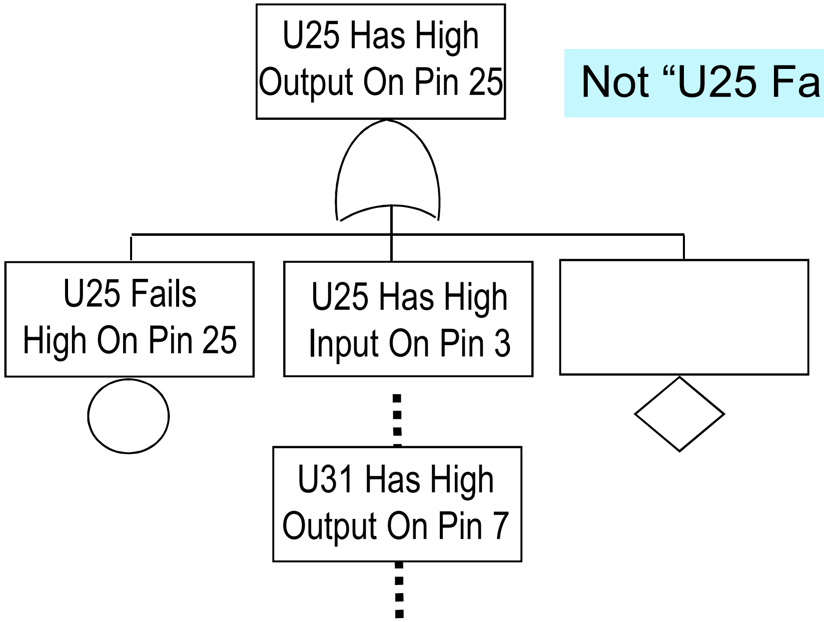
10 - Correct Node Wording Is Important

- 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





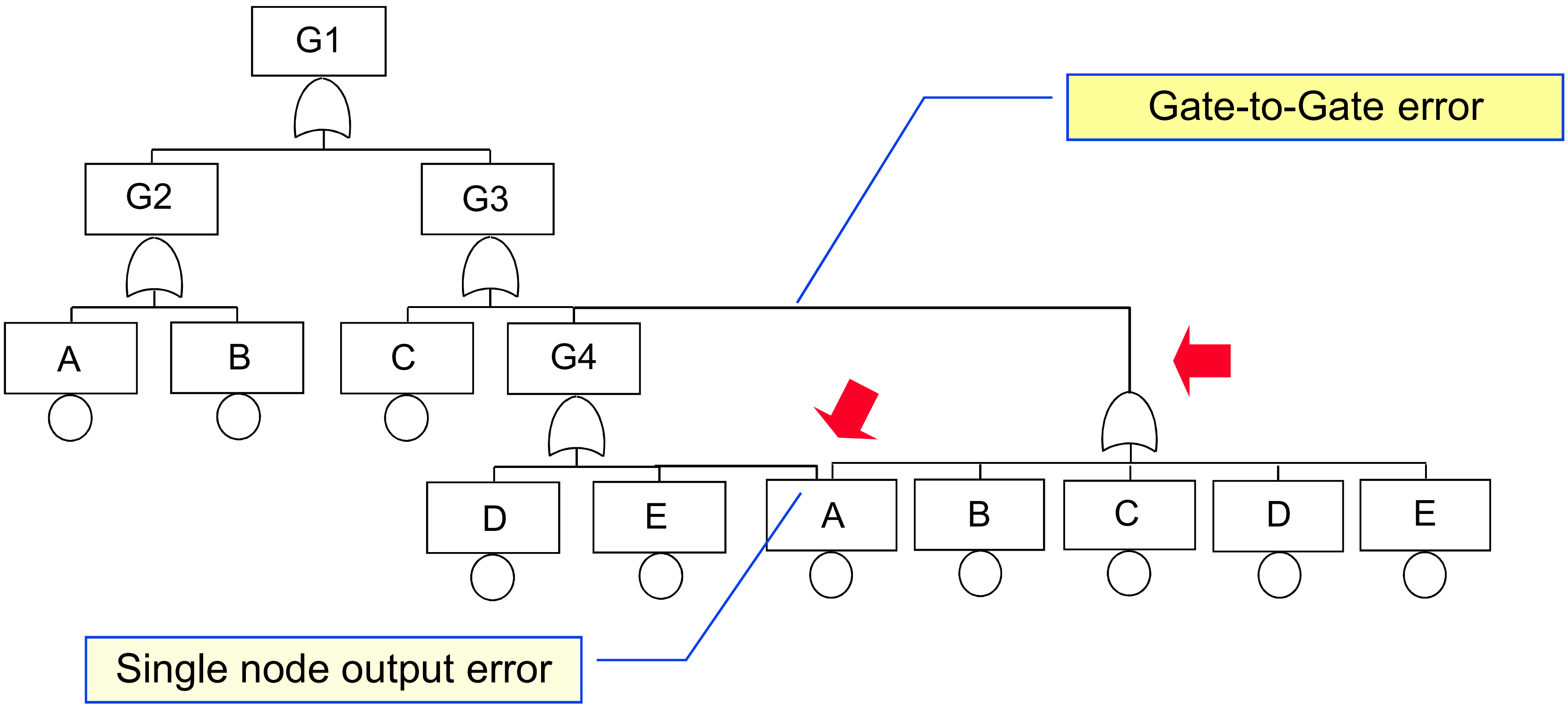
Not “U25 Fails”

Proper wording enhances the logic process

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

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. Its 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)

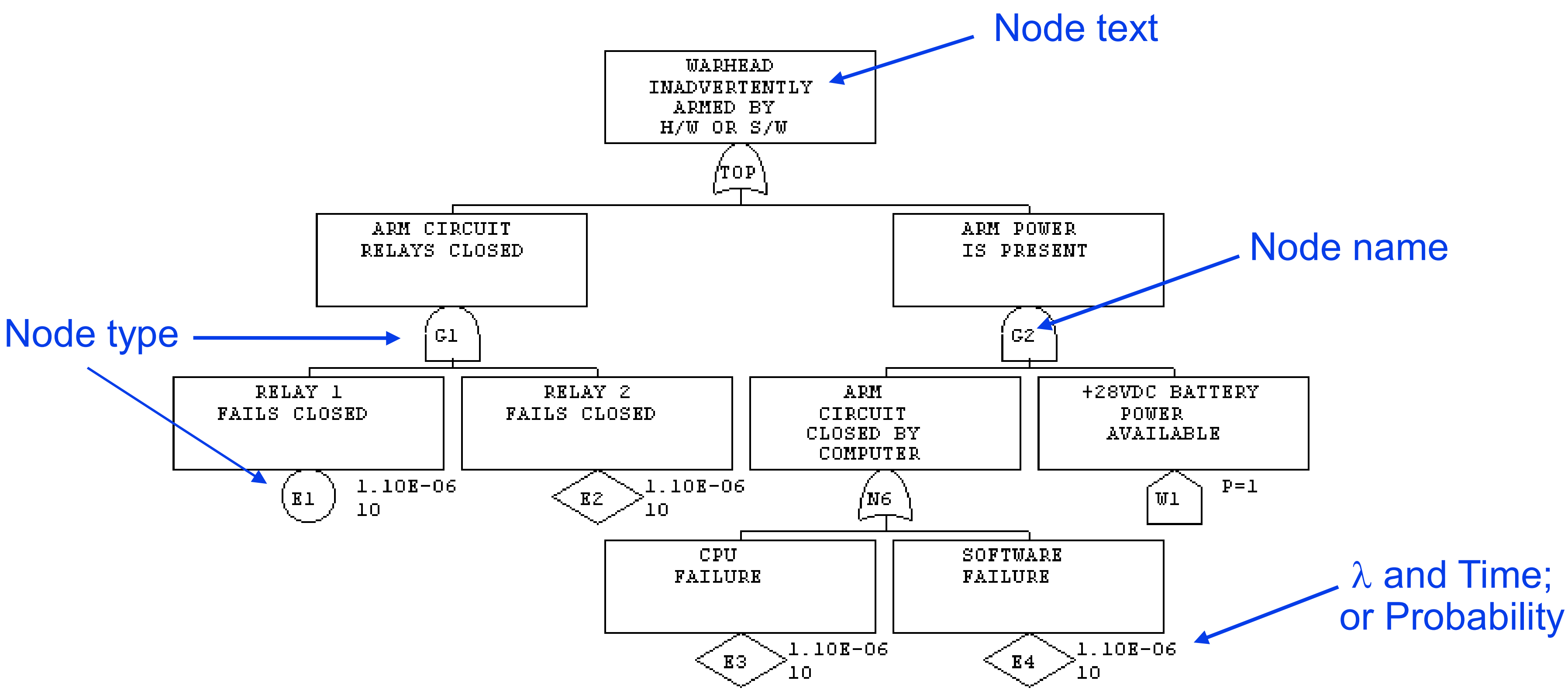
- Operator error should be included in the analysis where appropriate
  - It is up to the analyst and the purpose/objective of the FTA as to whether the event should be included in quantitative evaluations
  - The decision needs to be documented in the analysis ground rules

- 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

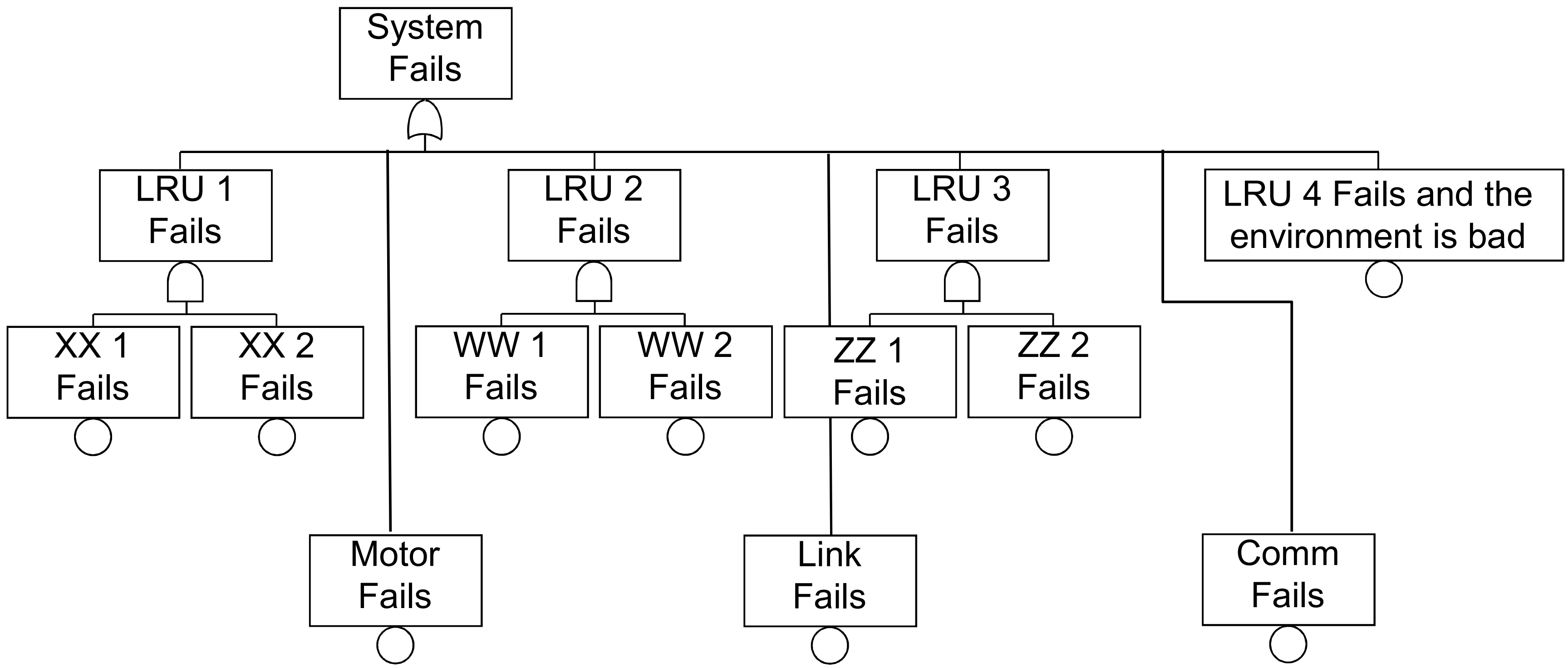


13 - Apply FT Aesthetics

- 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



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 (2.0×10^{-8} is as good as 1.742135×10^{-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

21 – The Human Is A System Element

- 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

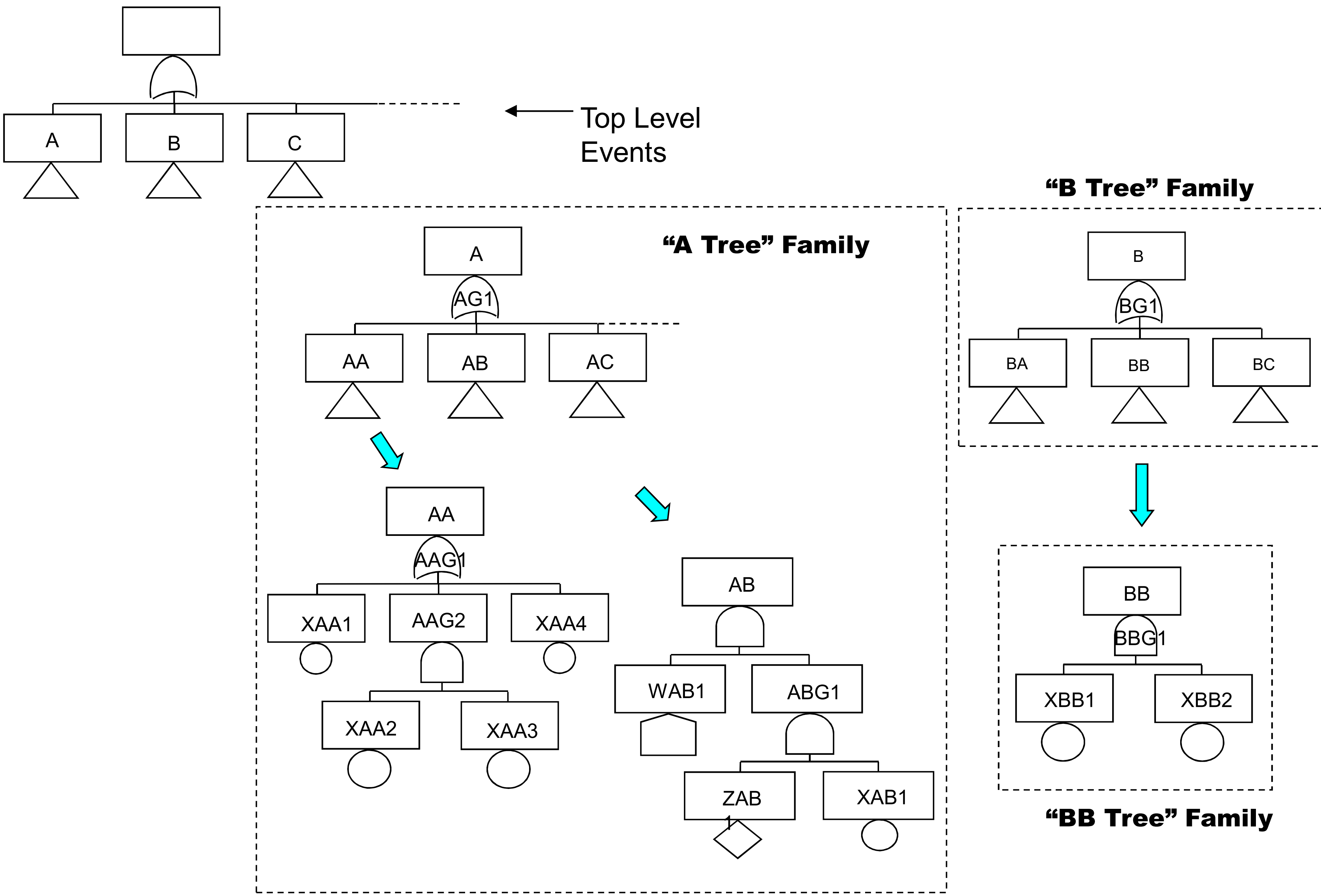
Sample Coding Scheme

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

1st Char	Symbol Type
G	Gate
X	Circle (primary failure)
Z	Diamond (secondary failure)
W	House (normal event)
Y	Oval (condition event)

Chars	Family	Represents
A	Top level family	Computer System
B	Top level family	Navigation System
AA	Member of A	
BB	Member of B	
ABC	Member of AB	

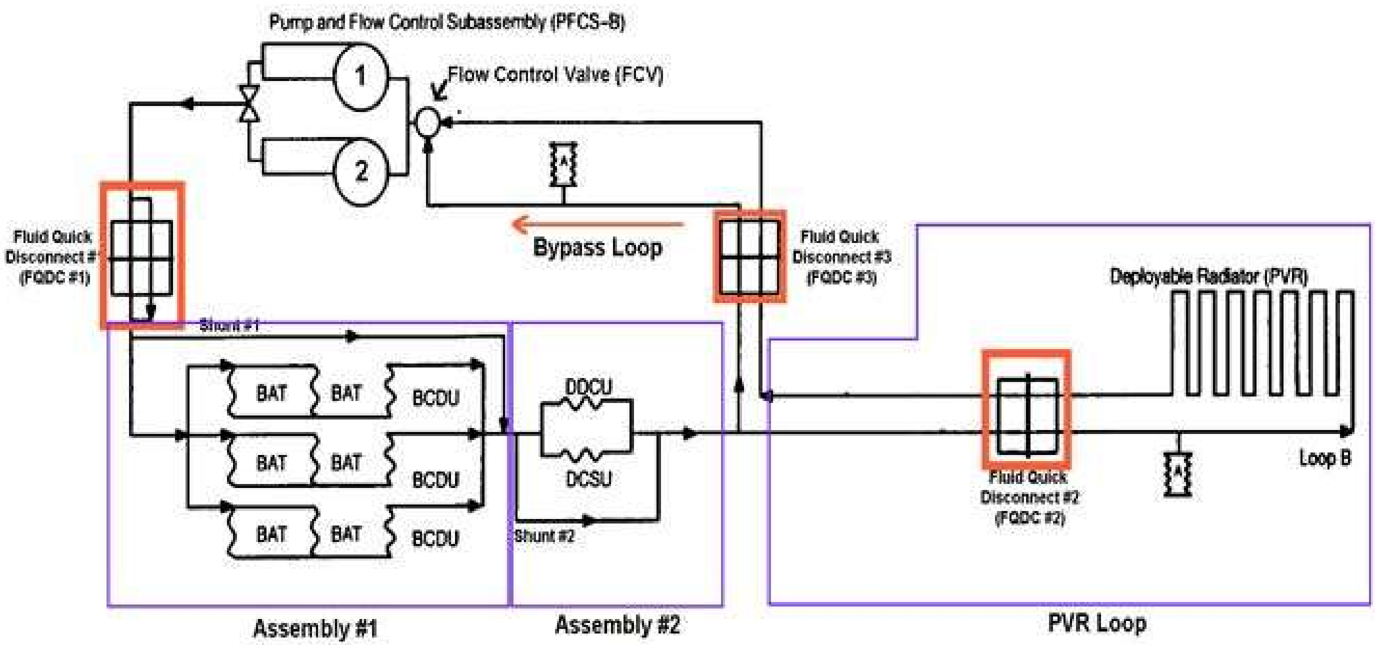
Coding Example

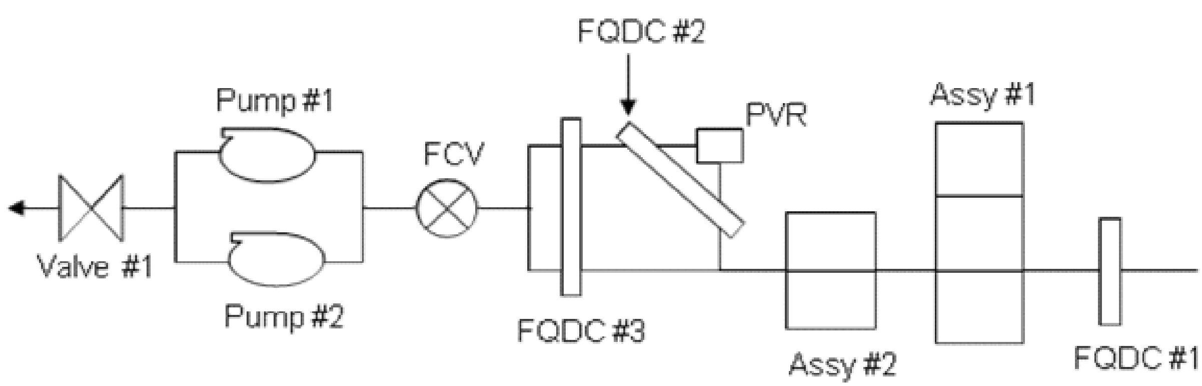


23 - FT Accounting

- 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





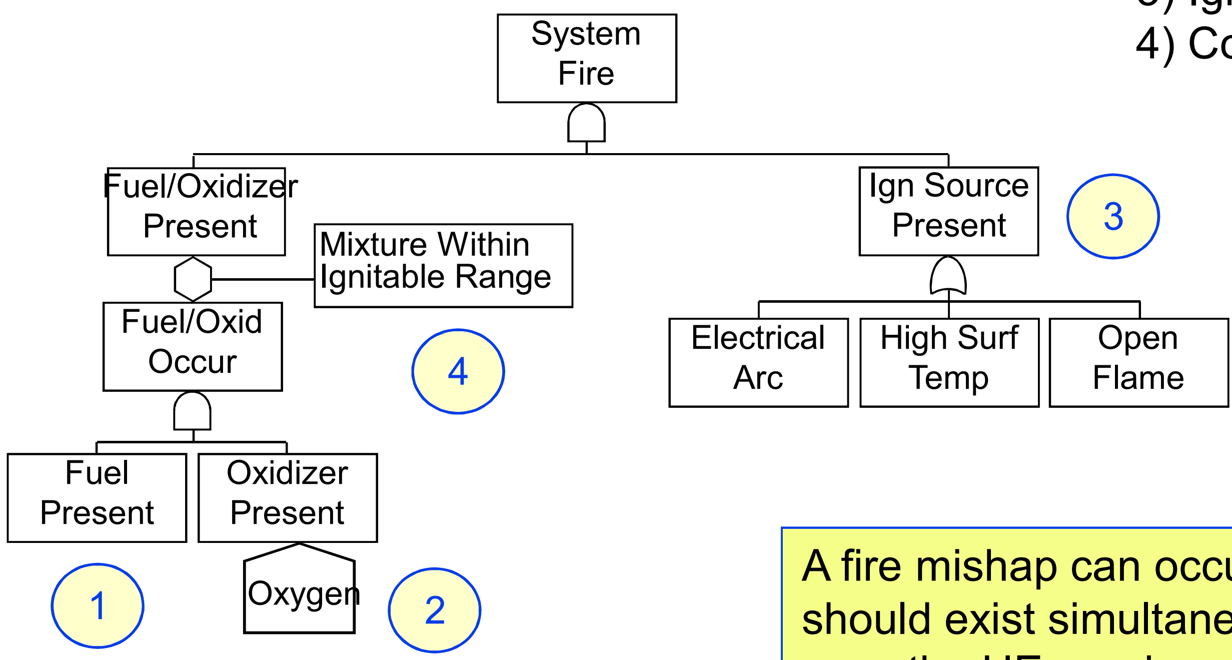
--- Example FTs of System Designs ---

System Scenario

System Design:

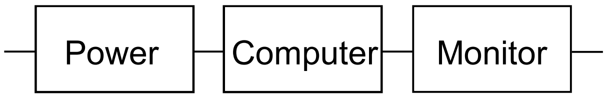
A system is comprised of fuel, oxidizer and an ignition source. When properly applied they run an engine.

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.

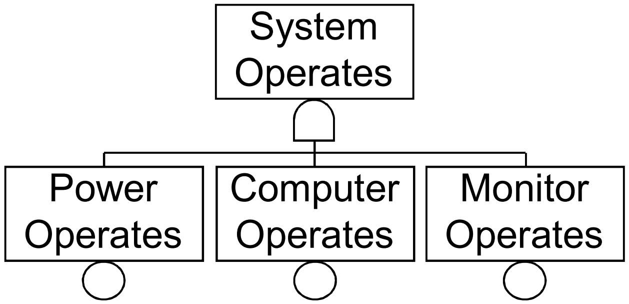
Series System



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

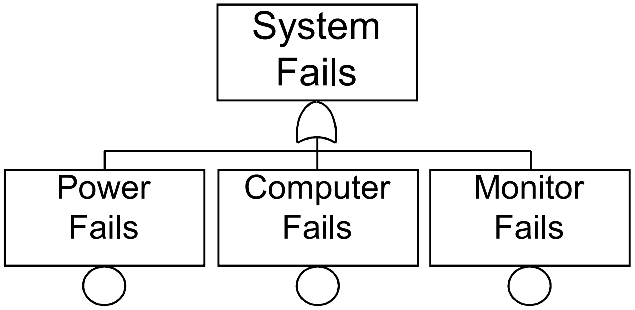
A ST is often the inverse of a FT

System Success

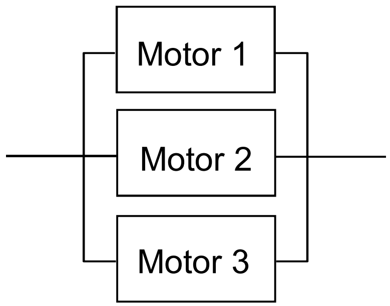


Success Tree (ST)

System Failure



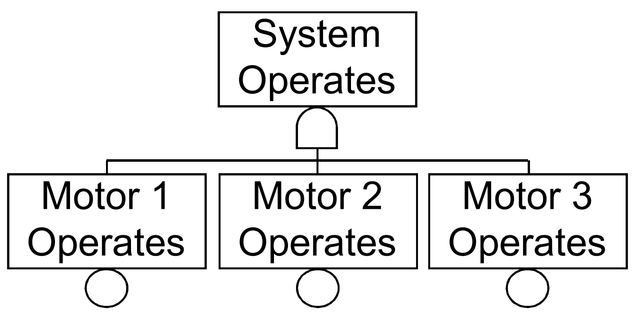
Parallel System



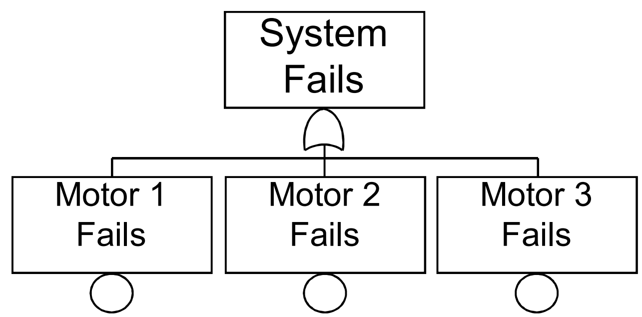
Case 1

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

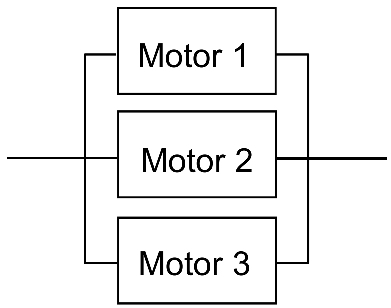
System Success



System Failure



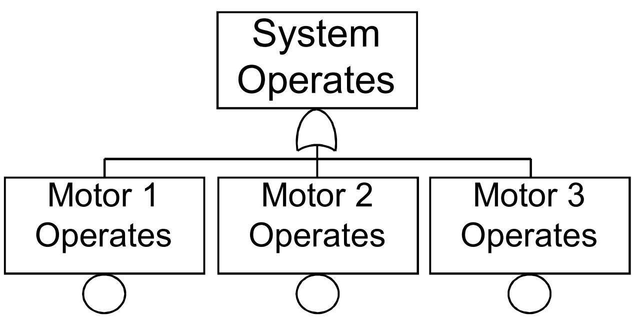
Parallel System



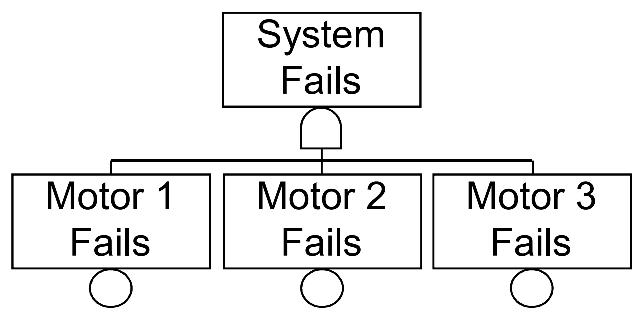
Case 2

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

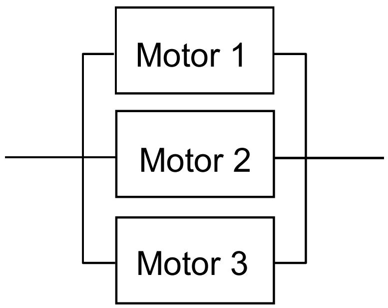
System Success



System Failure

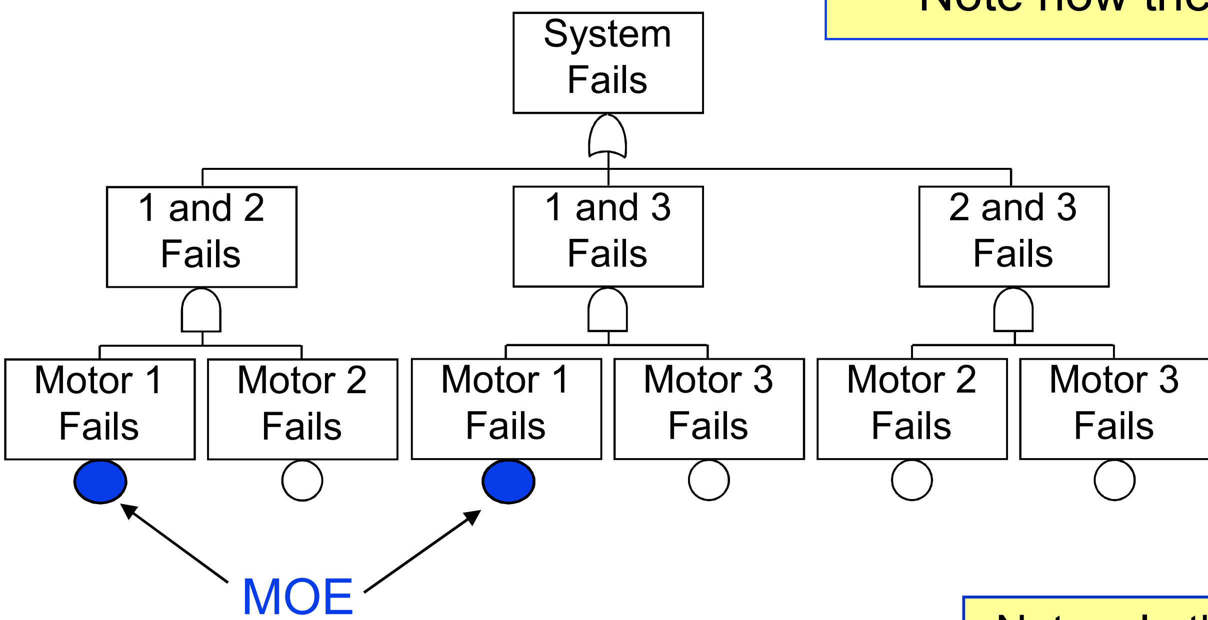


Parallel System



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.

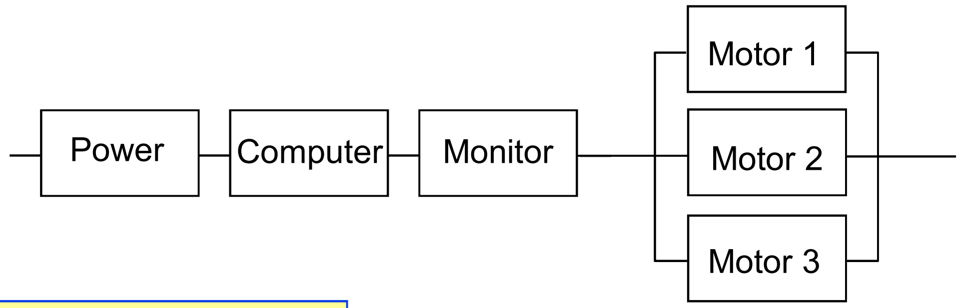


Note – In this case, If the FT is an inverted ST, the FT looks incorrect, but does work mathematically.

Series-Parallel System

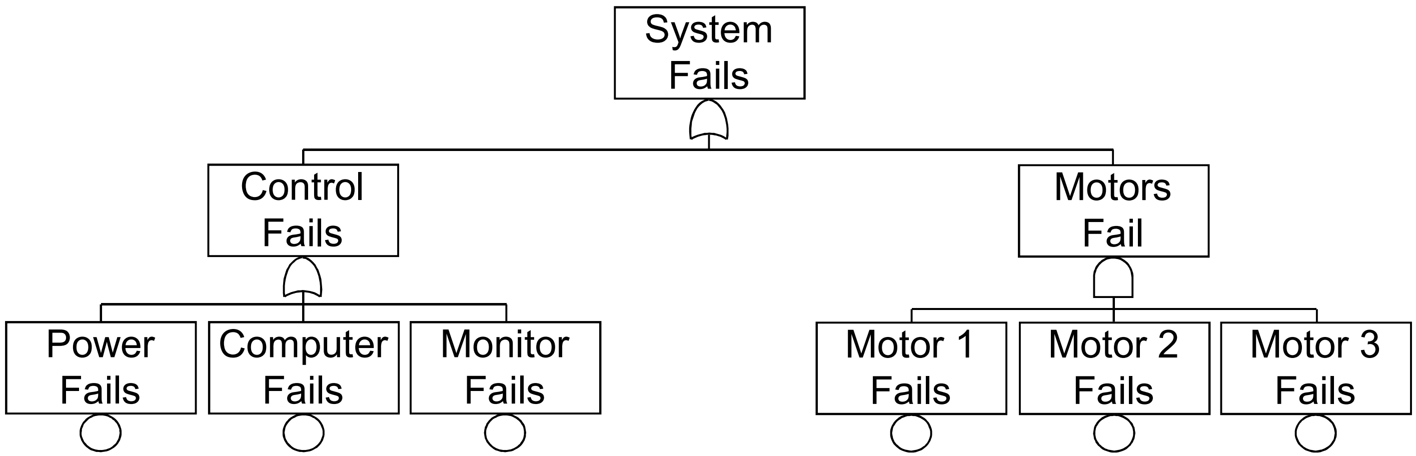
System Design:

A system is comprised of multiple components in series and parallel.



Therefore:

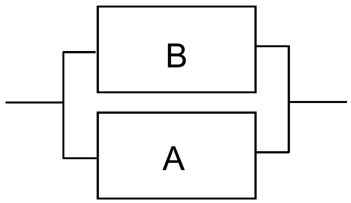
- System success requires that all series components must operate and at least one parallel component must operate successfully
- system failure occurs if one or more series components fail, or all parallel components fail

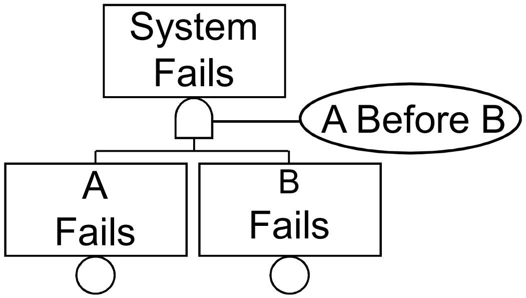


Sequence Parallel System

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 <u>only if A fails before B</u>.





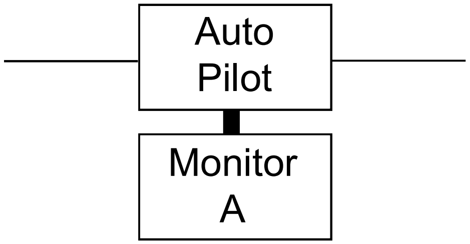
Therefore:

- Both must fail
- Sequential problem (A before B)
- The fault state logic requires a <u>Priority AND</u> 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:

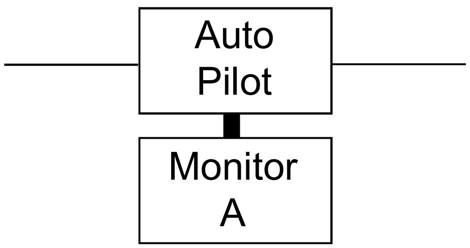
1. Full Monitor (full coverage)
2. Partial Monitor (partial coverage)

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

Monitor System

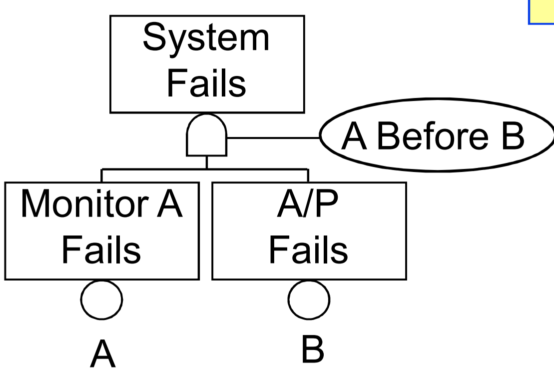
Case 1 – Full Monitor

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:

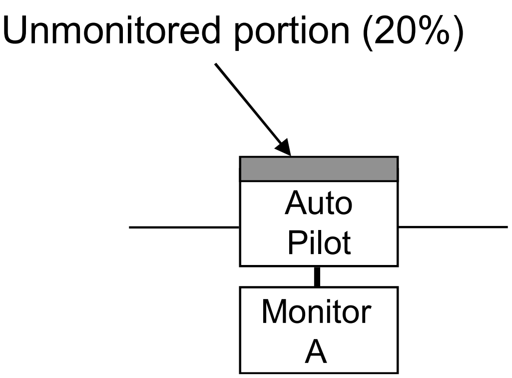
- Both must fail
- Sequential problem (A before B)
- The fault state logic requires a Priority AND gate



Monitor System

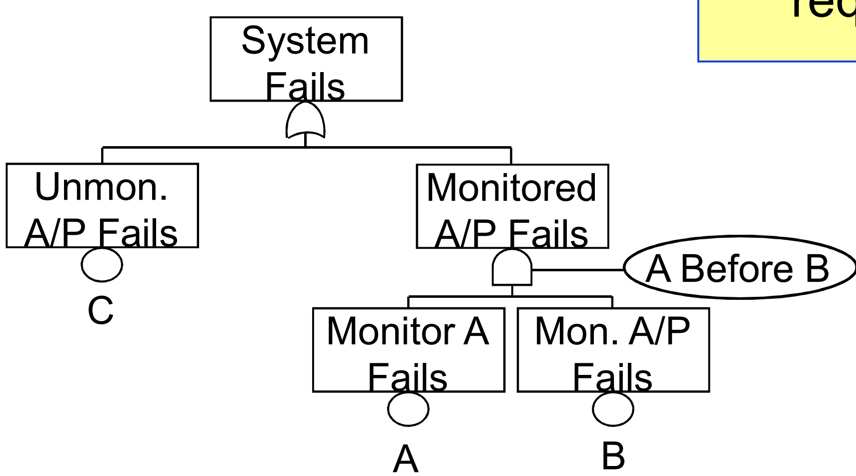
Case 2 – Partial Monitor

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



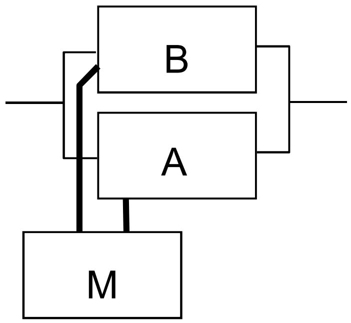
Standby System

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:

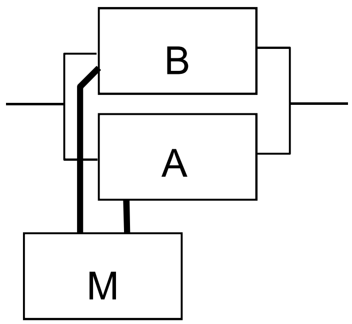
1. Hot Standby - powered during standby (uses operational λ_O)
2. Warm Standby - partially powered during standby ($\lambda_W < \lambda_O$)
3. Cold Standby - un-powered during standby ($\lambda_C = 0$)

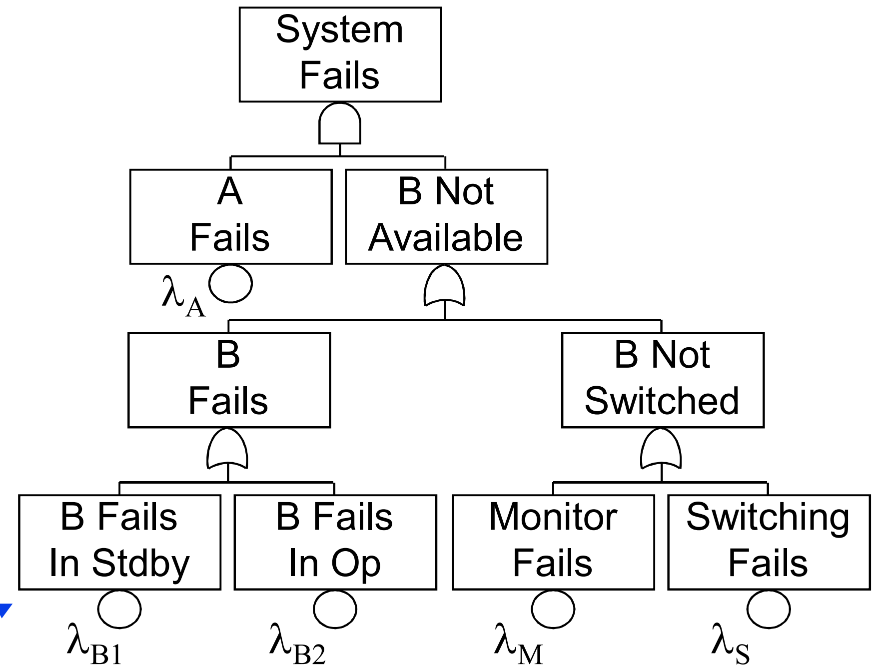


The major difference between each of these modes is that the failure rate is different in the standby mode.

Standby System

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

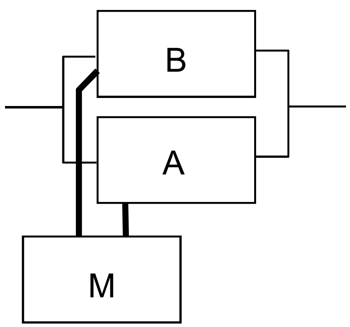


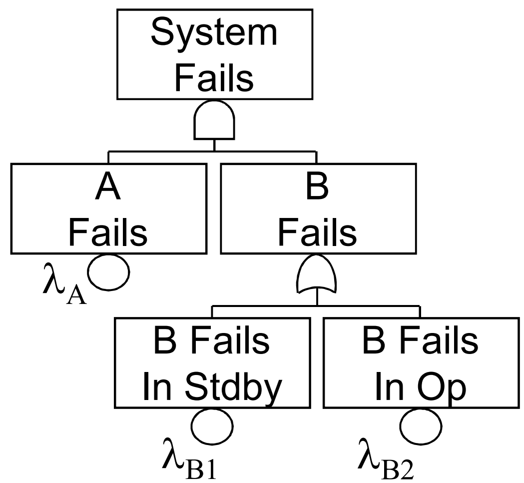


Standby System

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

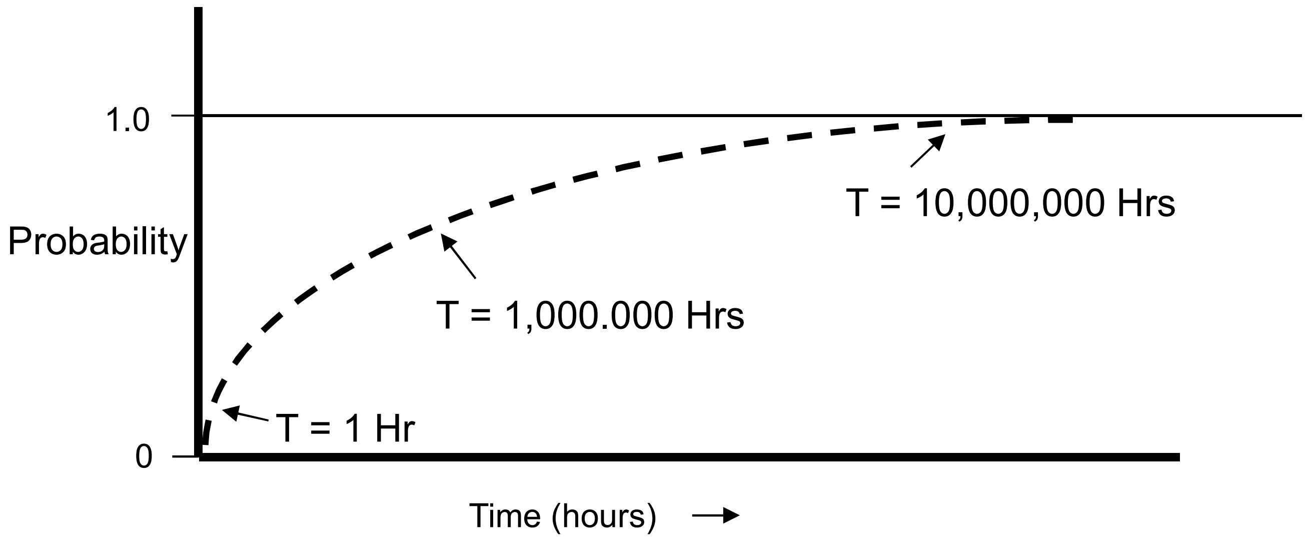
- $R = e^{-\lambda T}$
- $R + Q = 1$
- $Q = 1 - R = 1 - e^{-\lambda T}$
  - $Q \approx \lambda T$ when $\lambda T < 0.001$ (approximation)

Main equation of FTA

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

Effects 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



$\lambda = 1.0 \times 10^{-6}$

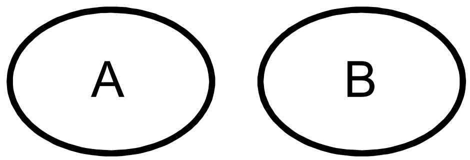
$P = 1 - e^{-\lambda T}$

The Effect of Exposure Time on Probability is Significant

Probability

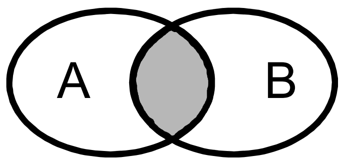
<u>Union (OR Gate)</u>

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.



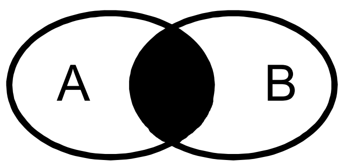
Case 1 - Disjoint Events

P=P(A) + P(B)



Case 2 - Non Disjoint Events

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



Case 3 - Mutually Exclusive Events

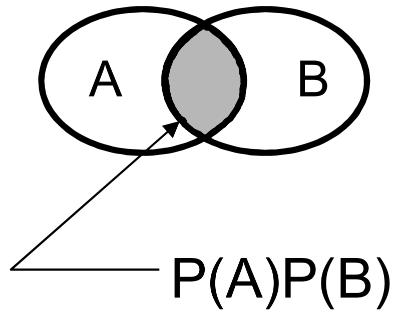
P=P(A) + P(B) - 2P(A)P(B)

Note - Exclusive OR is not the same as Disjoint.

Probability

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

$P = \Sigma(\text{singles}) - \Sigma(\text{pairs}) + \Sigma(\text{triples}) - \Sigma(\text{fours}) + \Sigma(\text{fives}) - \Sigma(\text{sixes}) + \bullet\bullet\bullet$

CS {A; B; C; D}

$$
\begin{aligned}
P = \quad & (P_A + P_B + P_C + P_D) \\
& - (P_{AB} + P_{AC} + P_{AD} + P_{BC} + P_{BD} + P_{CD}) \\
& + (P_{ABC} + P_{ABD} + P_{ACD} + P_{BCD}) \\
& - (P_{ABCD})
\end{aligned}
$$

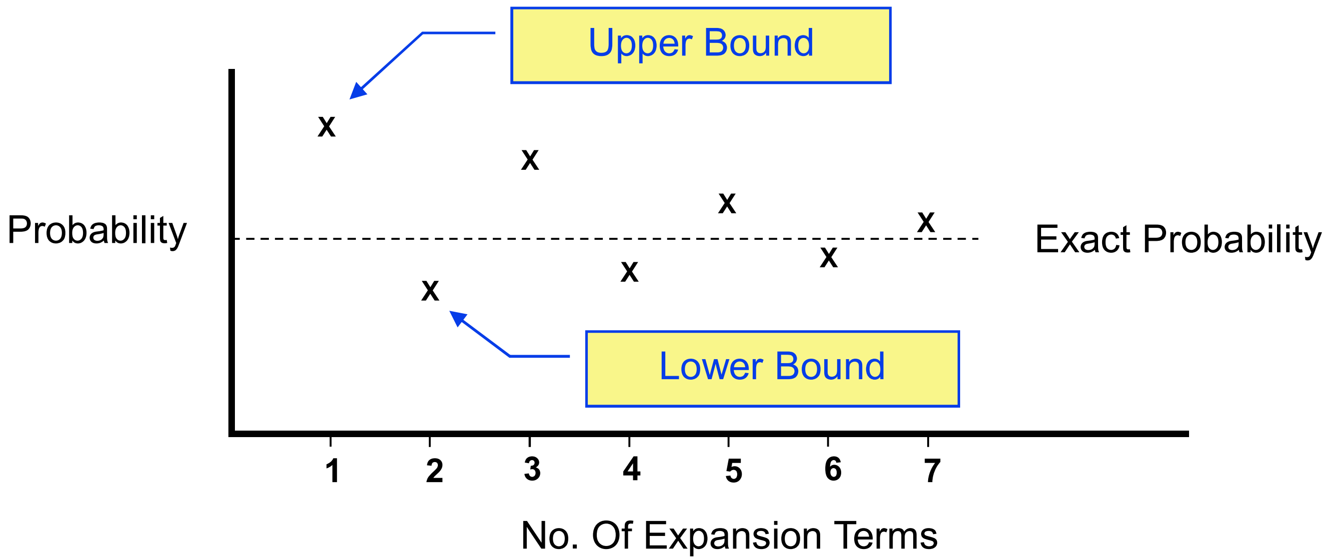
$$P = P_A + P_B + P_C + P_D - (P_{AB} + P_{AC} + P_{AD} + P_{BC} + P_{BD} + P_{CD}) + (P_{ABC} + P_{ABD} + P_{ACD} + P_{BCD}) - (P_{ABCD})$$

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}

$$P = \underbrace{P_A + P_B + P_C + P_D}_{\text{1st Term (all singles)}} - \underbrace{(P_{AB} + P_{AC} + P_{AD} + P_{BC} + P_{BD} + P_{CD})}_{\text{2nd Term (all doubles)}} + \underbrace{(P_{ABC} + P_{ABD} + P_{ACD} + P_{BCD})}_{\text{3rd Term (all triples)}} - \underbrace{(P_{ABCD})}_{\text{4th Term}} \cdots$$



Min Cut Set Upper Bound Approximation

$$P = 1 - [(1 - P_{CS1})(1 - P_{CS2})(1 - P_{CS3}).....(1 - P_{CSN})]$$

CS{A; B}

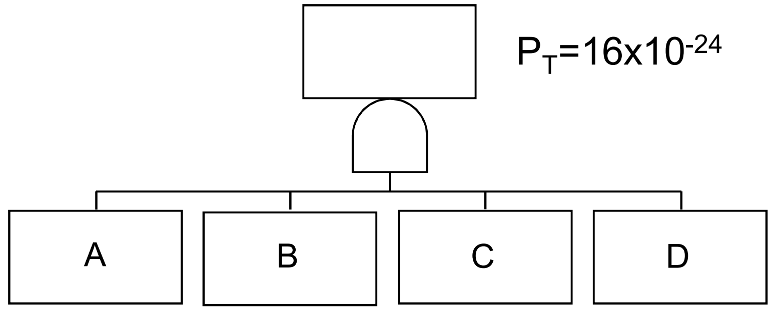
$$\begin{aligned} P &= 1 - [(1 - P_A)(1 - P_B)] \\ &= 1 - [1 - P_B - P_A + P_A P_B] \\ &= 1 - 1 + P_B + P_A - P_A P_B \\ &= P_A + P_B - P_A P_B \end{aligned}$$

Equivalent to standard expansion

FT Quantification

Note: AND gate reduces probability

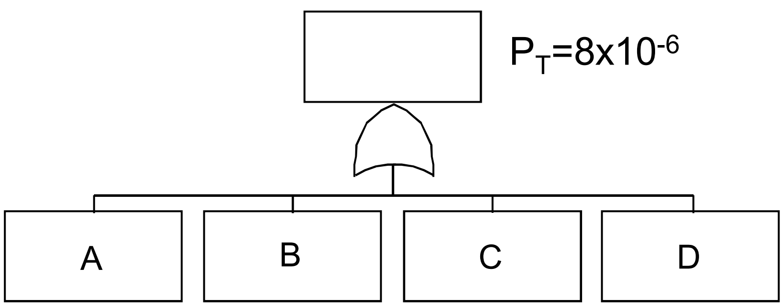
$P = P_A \bullet P_B \bullet P_C \bullet P_D$



$P_T = 16x10^{-24}$

$P_A = 2x10^{-6}$ $P_B = 2x10^{-6}$ $P_C = 2x10^{-6}$ $P_D = 2x10^{-6}$

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

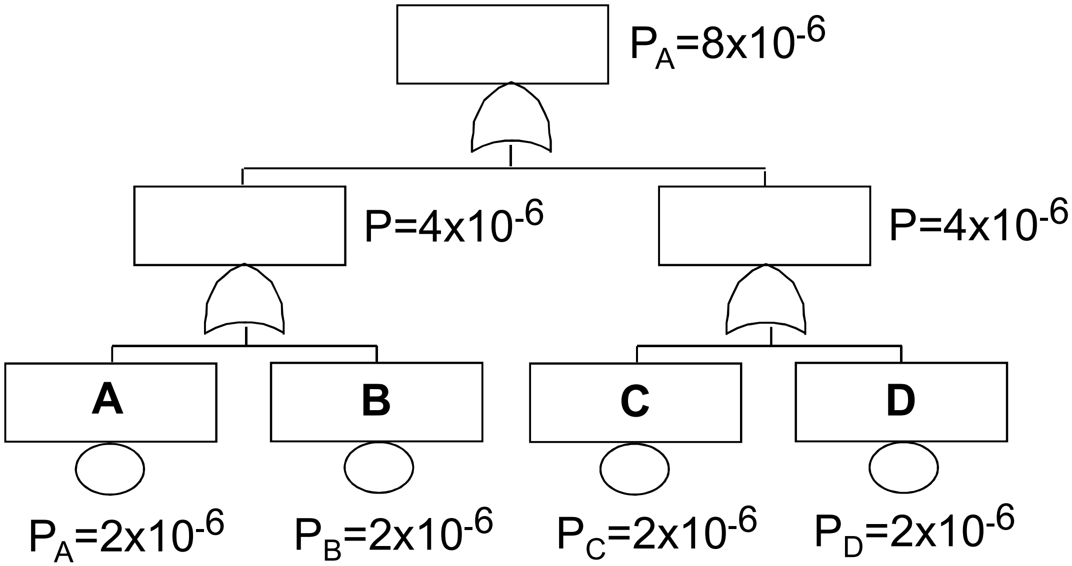


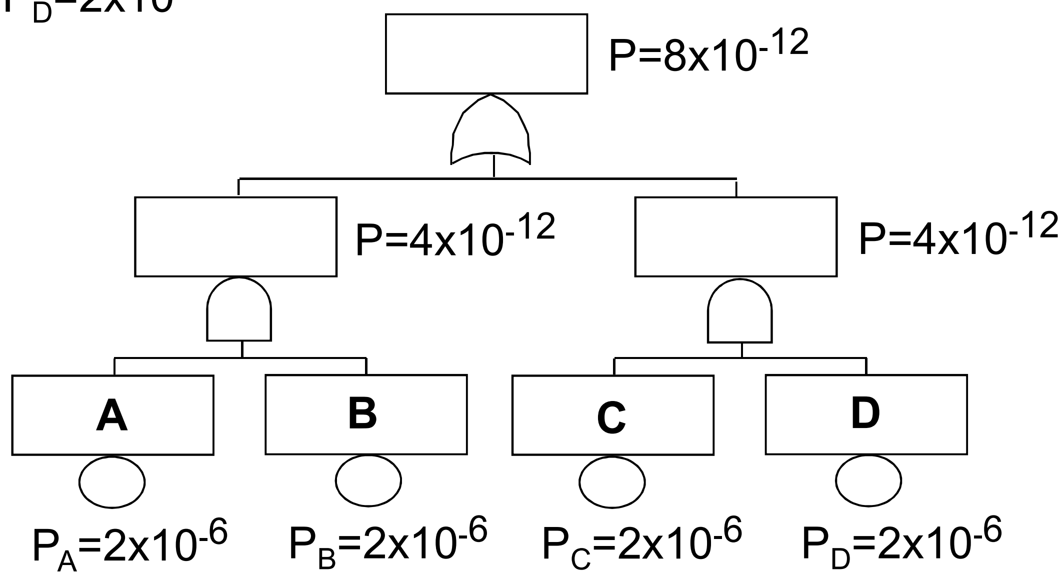
$P_T = 8x10^{-6}$

$P_A = 2x10^{-6}$ $P_B = 2x10^{-6}$ $P_C = 2x10^{-6}$ $P_D = 2x10^{-6}$

Note: OR gate increases probability and math complexity

FT Quantification





Axioms of Boolean Algebra

[A1]	$ab = ba$	Commutative Law
[A2]	$a + b = b + a$	
[A3]	$(a + b) + c = a + (b + c) = a + b + c$	Associative Law
[A4]	$(ab)c = a(bc) = abc$	
[A5]	$a(b+c) = ab + ac$	Distributive Law

Theorems of Boolean Algebra

[T1]	$a + 0 = a$	
[T2]	$a + 1 = 1$	
[T3]	$a \bullet 0 = 0$	
[T4]	$a \bullet 1 = a$	
[T5]	$a \bullet a = a$	
[T6]	$a + a = a$	✓ } Idempotent Law
[T7]	$a \bullet \overline{a} = 0$	✓ } Idempotent Law
[T8]	$a + \overline{a} = 1$	
[T9]	$a + ab = a$	
[T10]	$a(a + b) = a$	✓ } Law of Absorption
[T11]	$a + \overline{a}b = a + b$	✓ } Law of Absorption

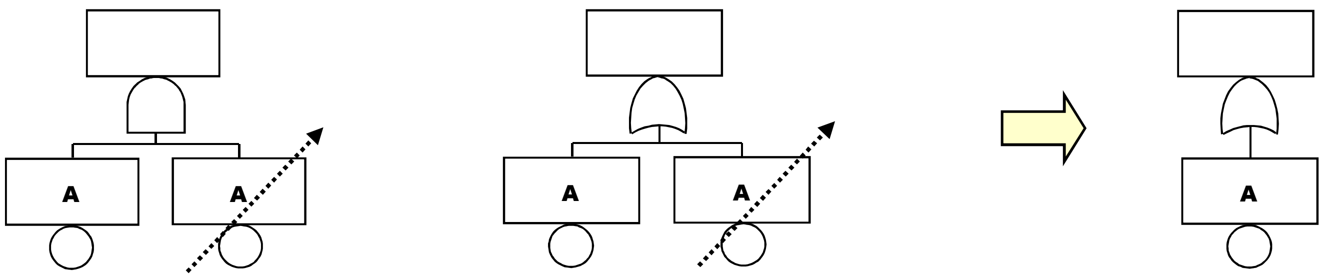
where $\overline{a}$ = not a

Example

[T5] $a \bullet a = a$ ✓

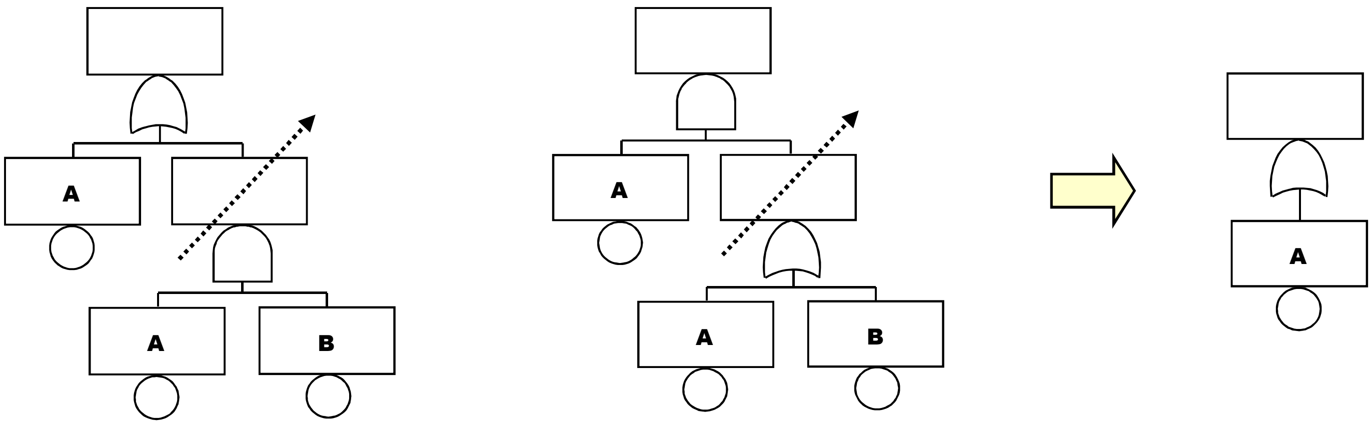
[T6] $a + a = a$ ✓

} Idempotent Law



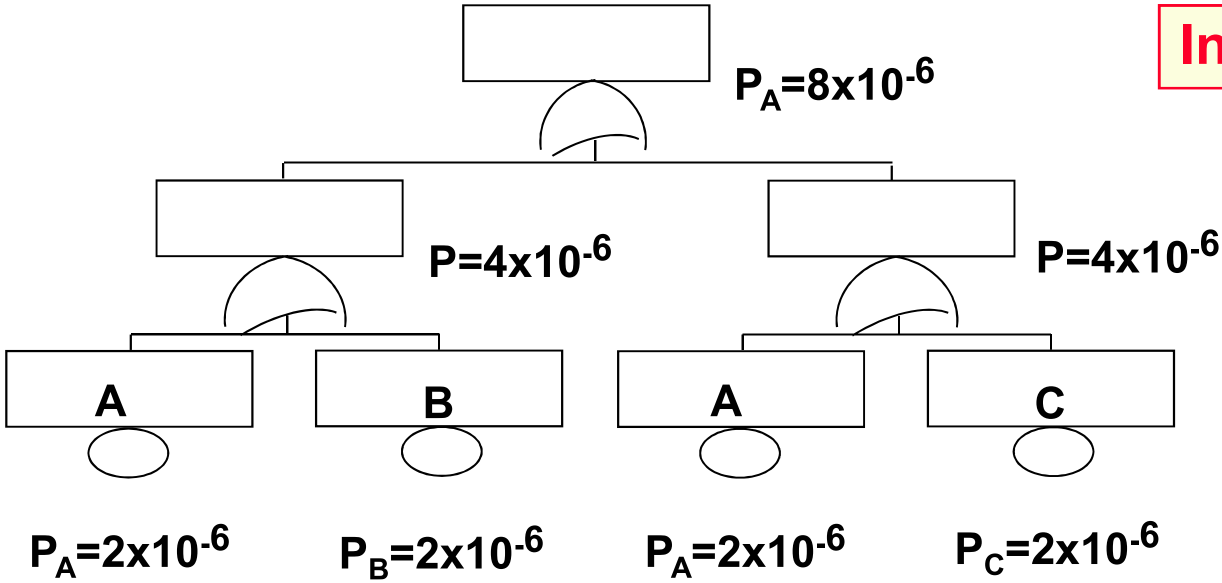
Example

[T9]	$a + ab = a$	✓	Law of Absorption
[T10]	$a(a + b) = a$	✓	

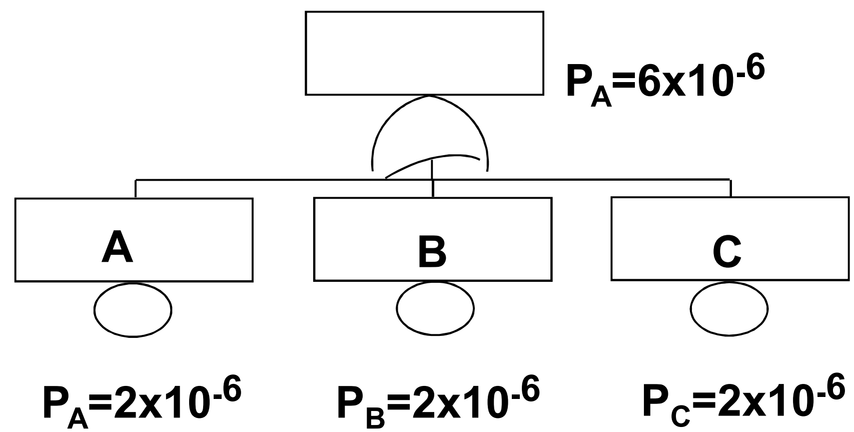


MOE Error Example 1

Incorrect



Correct

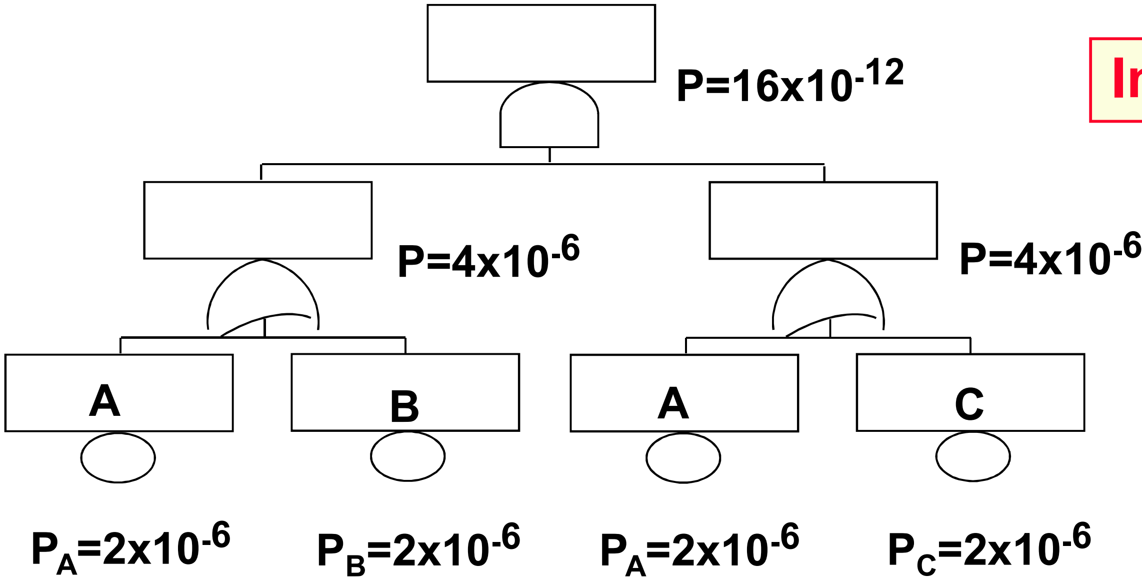


Cut Sets = A ; B ; C

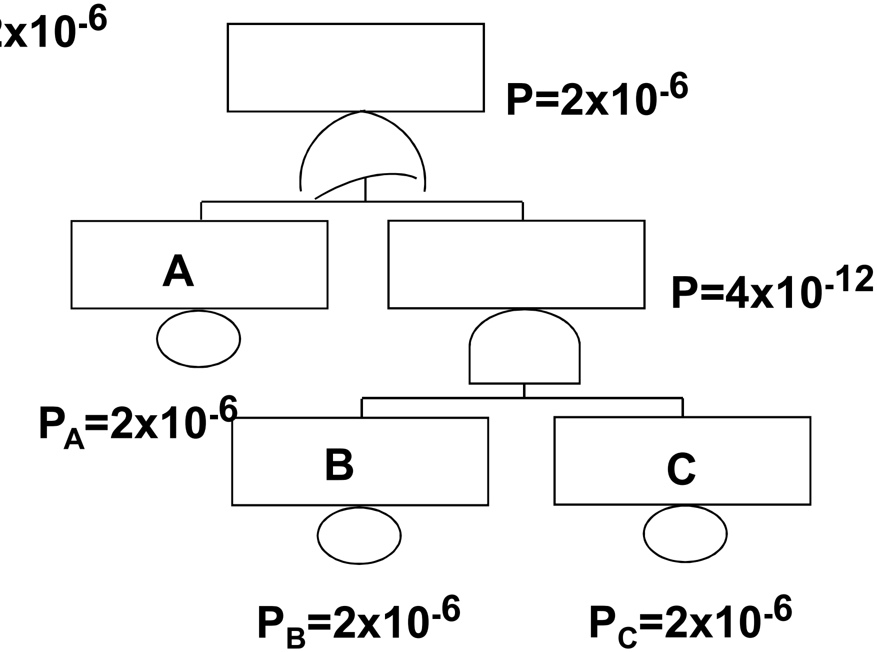
$P = P_A + P_B + P_C$
$= (2x10^{-6}) + (2x10^{-6}) + (2x10^{-6})$
= 6x10-6 [upper bound]

MOE Error Example 2

Incorrect



Correct

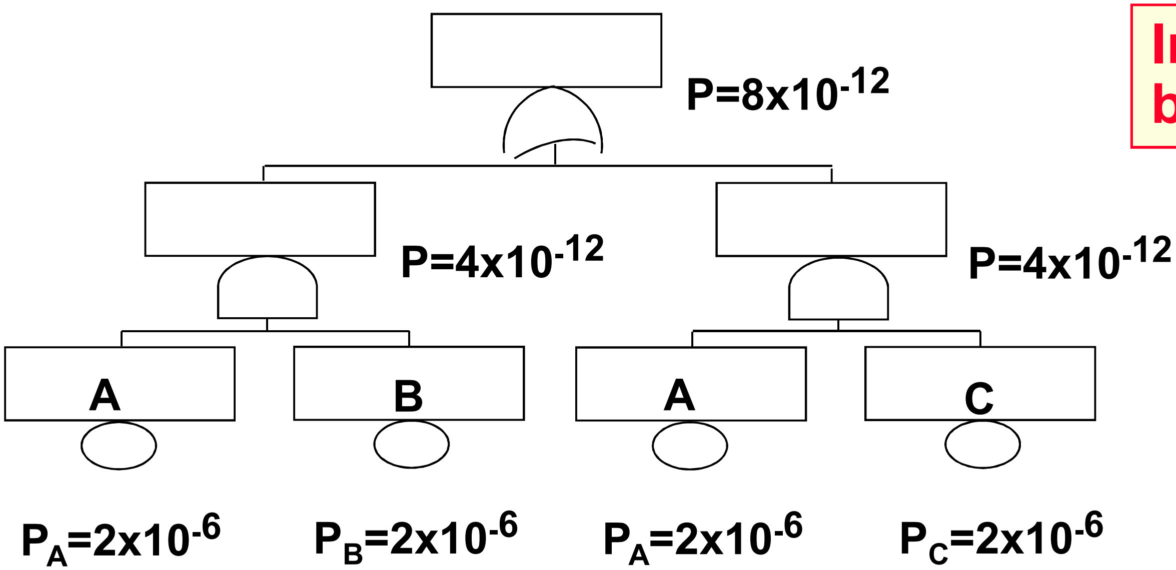


Cut Sets = A ; B,C

$P = P_A + P_B P_C$
$= (2x10^{-6}) + (2x10^{-6})(2x10^{-6})$
$= 2x10^{-6} + 4x10^{-12}$
= 2x10-6 [upper bound]

MOE Error Example 3

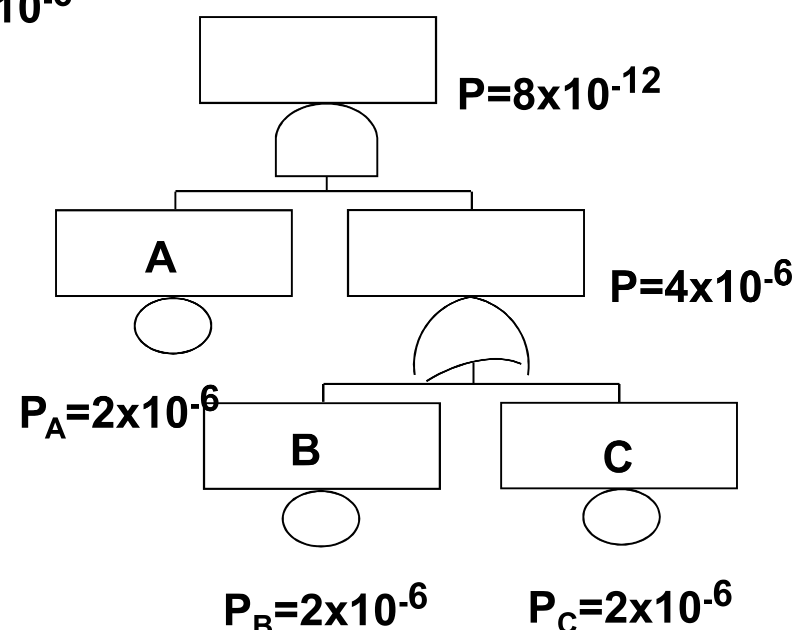
Incorrect
but accidentally Correct



Cut Sets = A,B ; A,C

$P = P_A P_B + P_A P_C$
$= (2x10^{-6})(2x10^{-6}) + (2x10^{-6})(2x10^{-6})$
$= 4x10^{-12} + 4x10^{-12}$
= 8x10-12 [upper bound]

Correct



--- 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

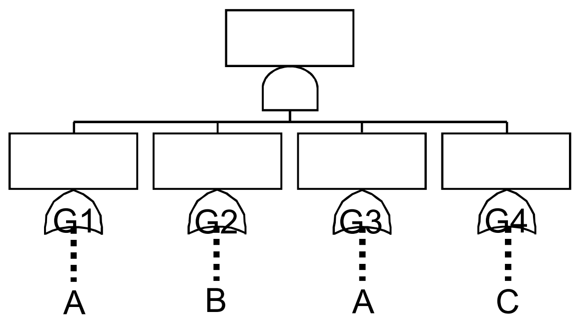
Solutions: 1) Prune FT, 2) Truncate FT or 3) FT Simulation

CS Truncation

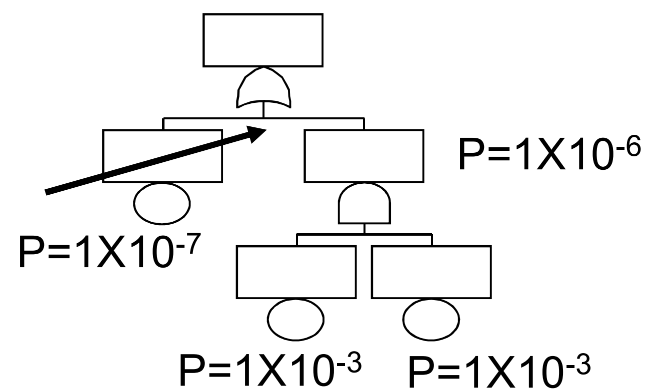
- Reduces number of CS's when tree is too large or complex
- Order Truncation
  - Throw away all CS's having more elements than order N_{CO}
  - Example – if N_{CO} is 3, then CS{A, B, C, D} would be dropped
- Probability Truncation
  - Throw away all CS's having probability smaller than P_{CP}
  - Example – if P_{CP} is 1.0×10^{-6}, then CS(1.0×10^{-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

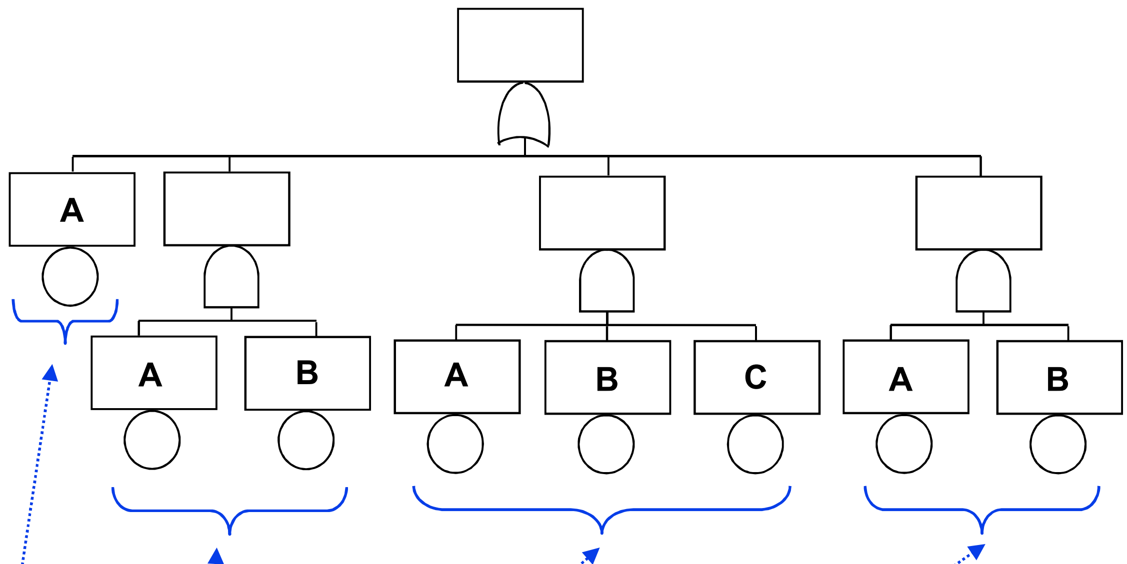


Watch for SPFs

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)

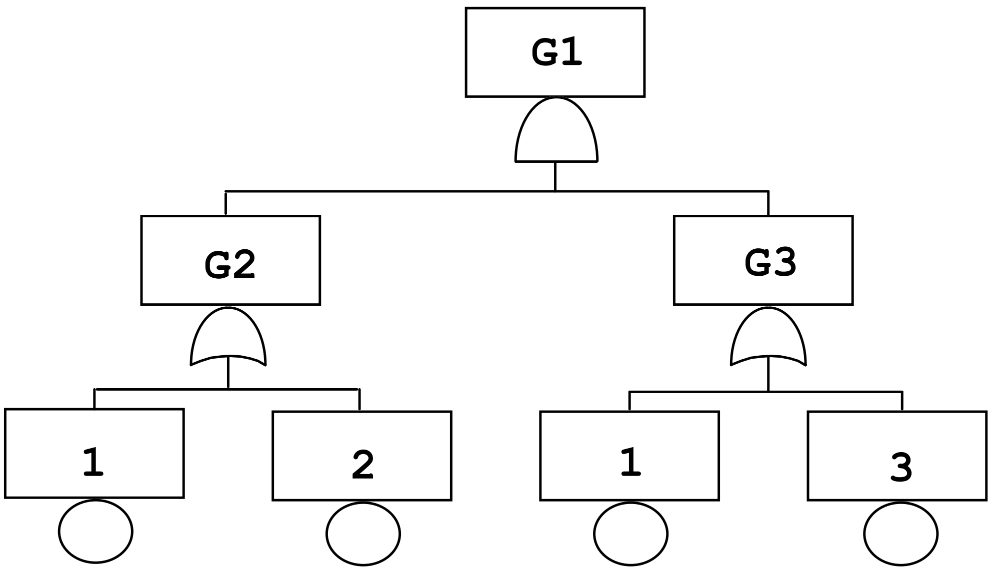
Min CS



Cut Sets:
A
A,B ← **SCS**
A,B,C ← **SCS**
A,B ← **DupCS, SCS**

Min Cut Sets:
A

MOCUS Algorithm



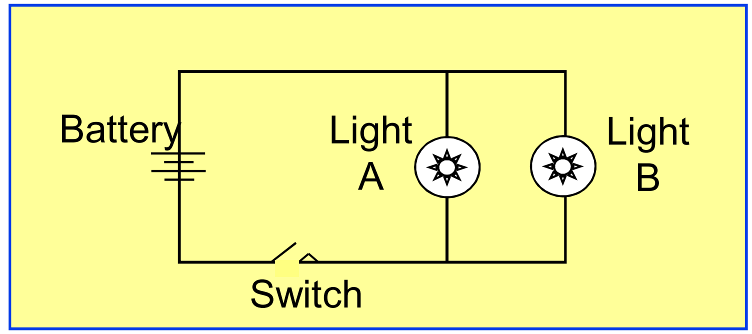
1	2	3	4	5	6
G1	G2,G3	1, G3	1, 1	1	1
			1, 3	1, 3	
		2, G3	2, 1	1, 2	
			2, 3	2, 3	2, 3

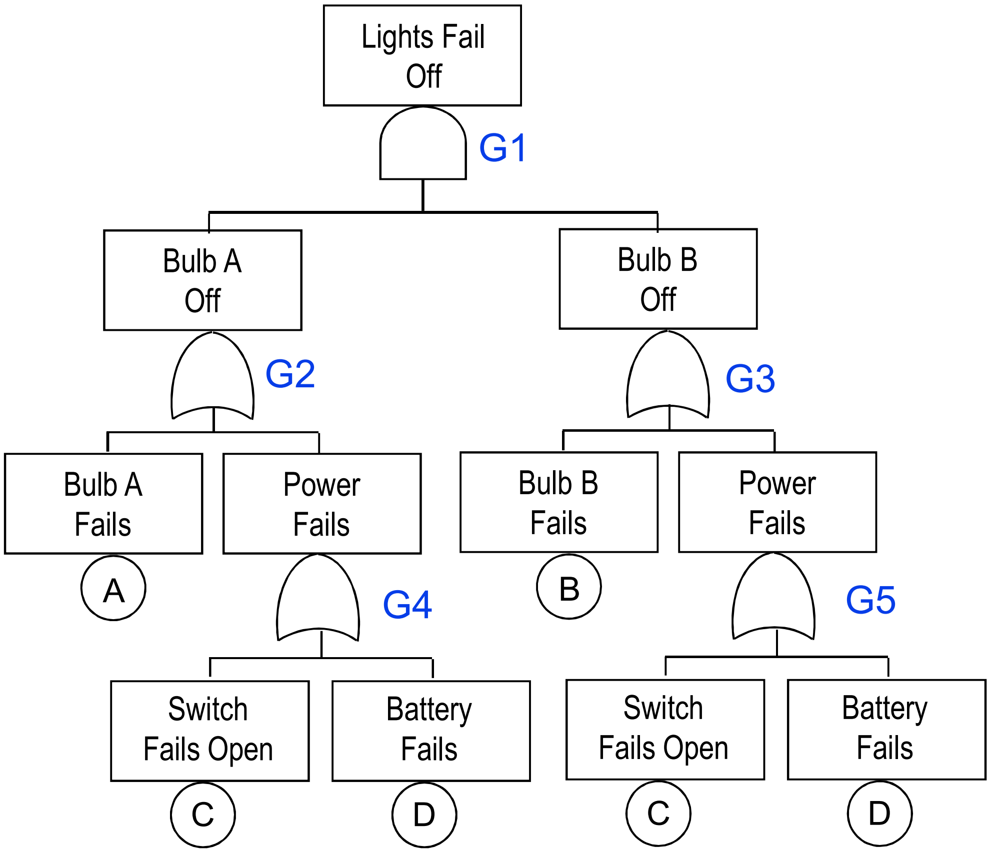
1	2	3	4	5	6
~~G1~~	~~G2~~,G3	1, ~~G3~~	1, ~~1~~	1	1
			1, 3	~~1~~, 3	
		2, ~~G3~~	2, 1	~~1~~, 2	
			2, 3	2, 3	2, 3

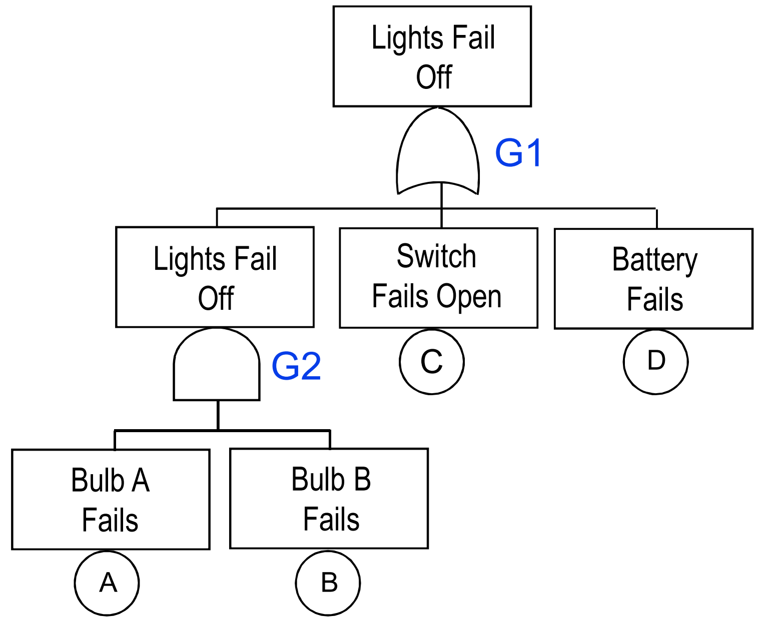
SuperCS MinCS

MOCUS - Method of Obtaining Cut Sets

Are these two FTs equal?





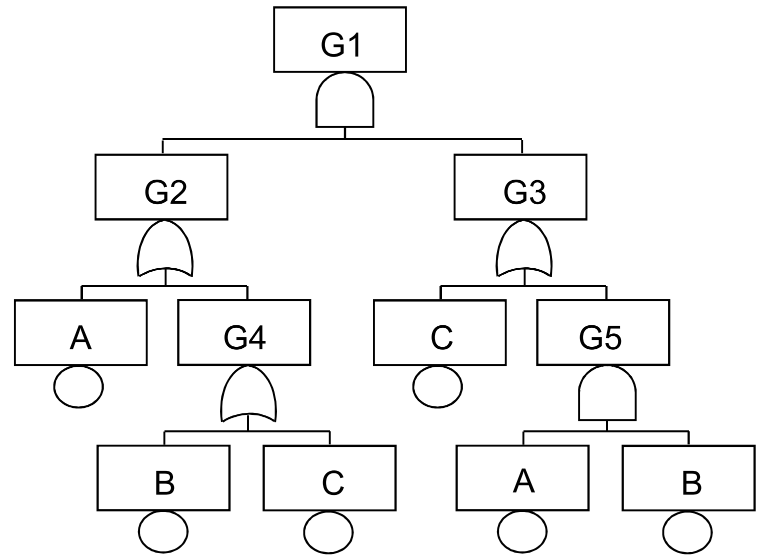


$G1=G2 \bullet G3$
$G1=(A+G4) \bullet G3$
$G1=(A+C+D) \bullet G3$
$G1=(A+C+D) \bullet (B+G5) = (A+C+D) \bullet (B+C+D)$
$G1=AB+AC+AD+CB+CC+CD+DB+DC+DD$
$G1=AB+AC+AD+CB+C+CD+DB+DC+D$
$G1=AB+C+D$

$G1=G2 + C + D$
$G1=AB+C+D$

Evaluation Example

Top Down Approach (MOCUS)



G1=G2 • G3

C
G5

AB

C+AB

A • G3
G4 • G3

B • G3
C • G3

G1=AB+AC

A • (C+AB) = AC +AAB = AC + AB

B • (C+AB) = BC +BAB = BC + AB

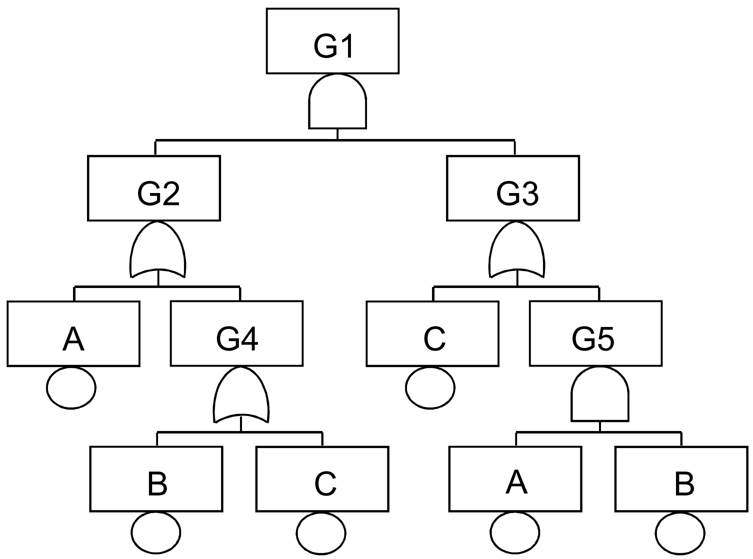
C • (C+AB) = CC +CAB = C + ABC

√ √ √ √

AC+AB + BC + AB + C + ABC

C + AB

Evaluation Example



Bottom Up Approach

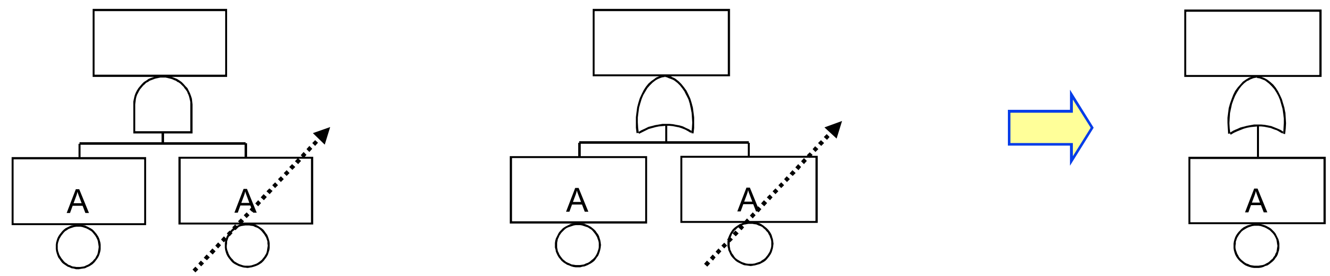
G5 = A,B
G3 = C + G5 = C + A,B
G4 = B + C
G2 = A + G4 = A + B + C
G1 = G2 • G3
= (A + B + C) (C + A,B)
= A,C + A,A,B + B,C + B,A,B + C,C + C,A,B
= A,C + A,B + B,C + A,B + C + A,B,C
= C + A,C + B,C + A,B + A,B,C
= **C + A,B**

Boolean Reductions

$a \bullet a = a$

$a + a = a$

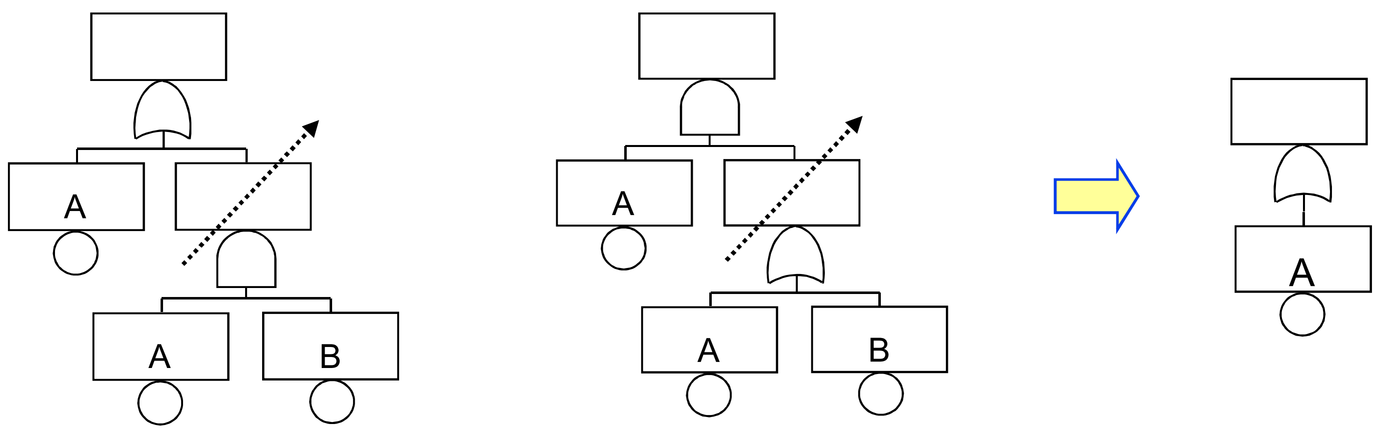
Idempotent Law



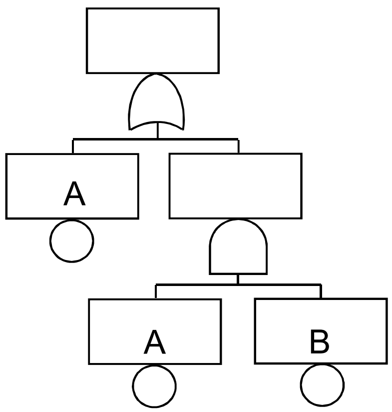
$a + ab = a$

$a(a + b) = a$

Law of Absorption

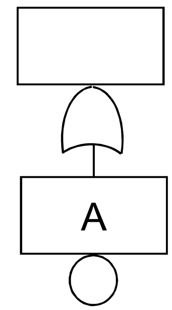


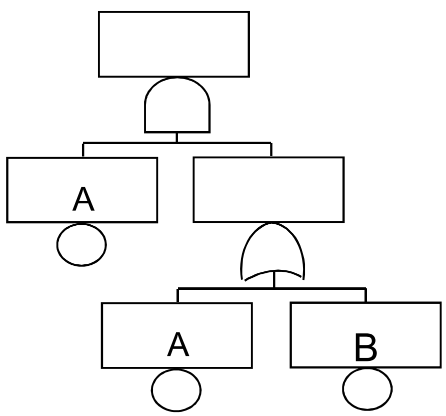
MOE Reduction



[1] A
[2] A,B

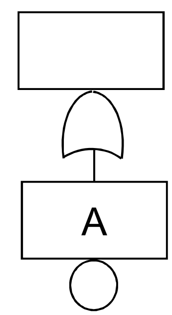
[1] A



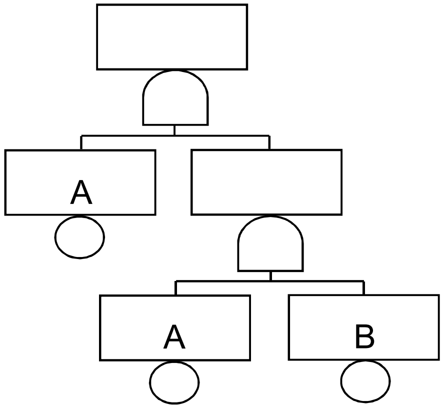


[1] A,A
[2] A,B

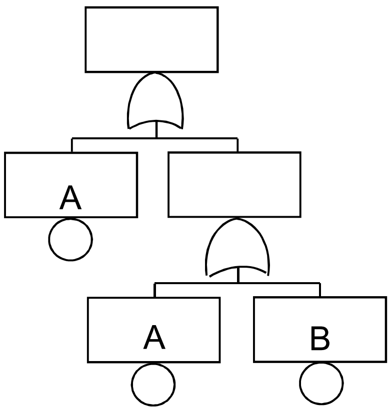
[1] A



MOE Reduction



[1] A,A,B → [1] A,B



[1] A
[2] A
[3] B

→

[1] A
[2] B

FT Approximations vs. Markov

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

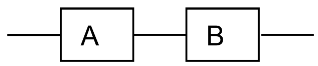
- FTA
  - Defined, structured and rigorous methodology
  - Easy to learn, perform and follow
  - Provides root causes
  - Displays cause-consequence relationships
  - Sufficient accuracy when approximations are used

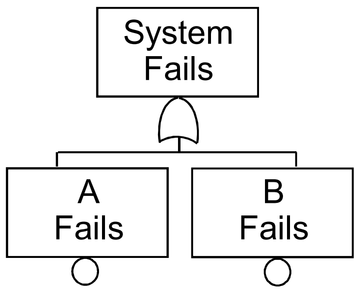
FTA is often criticized as not being accurate enough.

Series System

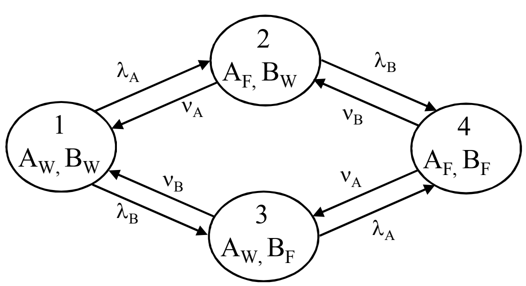
Description

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.





$$P = (1 - e^{-\lambda AT}) + (1 - e^{-\lambda BT}) - (1 - e^{-\lambda AT})(1 - e^{-\lambda BT})$$
$$= 1 - e^{-(\lambda A+-\lambda B)T}$$



$$dP_1 / dt = - (\lambda_A + \lambda_B)P_1 + \nu_A P_2 + \nu_B P_3$$
$$dP_2 / dt = \lambda_A P_1 - (\lambda_A + \nu_A)P_2 + \nu_B P_4$$
$$dP_3 / dt = \lambda_B P_1 - (\lambda_A + \nu_A)P_3 + \nu_A P_4$$
$$dP_4 / dt = \lambda_B P_2 + \lambda_A P_3 - (\nu_A + \nu_B)P_4$$
$$P = P_2 + P_3 + P_4$$
$$P = (1 - e^{-\lambda AT}) + (1 - e^{-\lambda BT}) - (1 - e^{-\lambda AT})(1 - e^{-\lambda BT})$$
$$= 1 - e^{-(\lambda A+-\lambda B)T}$$

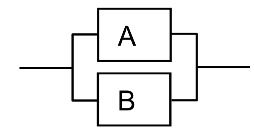
Conclusion

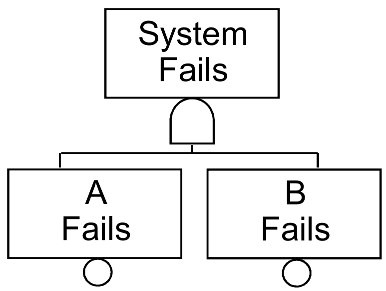
Both methods produce the same results (for non-repair case).

Parallel System

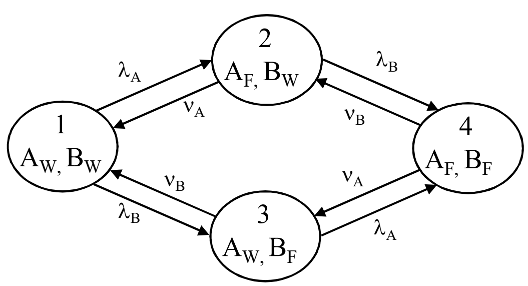
Description

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.





$$P = (1 - e^{-\lambda AT})(1 - e^{-\lambda BT})$$



$$dP_1 / dt = -(\lambda_A + \lambda_B)P_1 + \nu_A P_2 + \nu_B P_3$$
$$dP_2 / dt = \lambda_A P_1 - (\lambda_A + \nu_A)P_2 + \nu_B P_4$$
$$dP_3 / dt = \lambda_B P_1 - (\lambda_A + \nu_A)P_3 + \nu_A P_4$$
$$dP_4 / dt = \lambda_B P_2 + \lambda_A P_3 - (\nu_A + \nu_B)P_4$$
$$P = P_4$$
$$P = (1 - e^{-\lambda AT})(1 - e^{-\lambda BT})$$

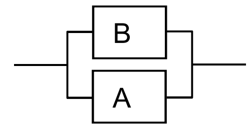
Conclusion

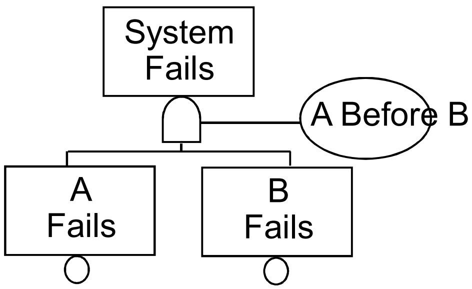
Both methods produce the same results (for non-repair case).

Sequence Parallel System

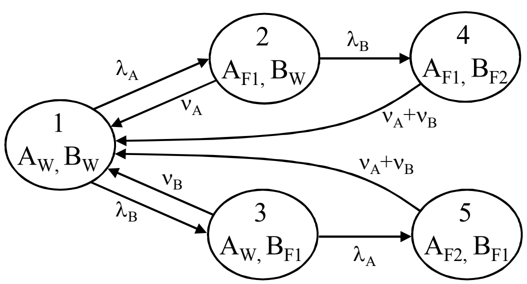
Description

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 if both fail, but only if A fails before B.





$$P = (P_A \bullet P_B) / N!$$
$$P = (P_A \bullet P_B) / 2$$
$$= ((1 - e^{-\lambda AT})(1 - e^{-\lambda BT})) / 2$$



$$P = \frac{\lambda_A(1 - e^{-\lambda BT}) - \lambda_B(e^{-\lambda BT} - e^{-(\lambda A+\lambda B)T})}{\lambda_A + \lambda_B}$$

Conclusion

Each method produces a different equation, but results are comparable.

Sequence Parallel System

Comparison of Results for Sequence Parallel System
Where λ_A=1.0x10^{-6} and λ_B=1.0x10^{-7}

Time (Hrs)	FTA	MA
1	5.00000E-14	5.00000E-14
10	4.99947E-12	4.99998E-12
100	4.99973E-10	4.99980E-10
1,000	4.99725E-8	4.99800E-8
10,000	4.97260E-6	4.98006E-6
100,000	4.73442E-4	4.80542E-4
1,000,000	3.00771E-2	3.45145E-2

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

Error	Consequence
■ High	Erroneous top probability
■ Medium	Insufficient Info; not sure if results are incorrect
■ Low	Poor FT diagram, however, results are likely correct

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

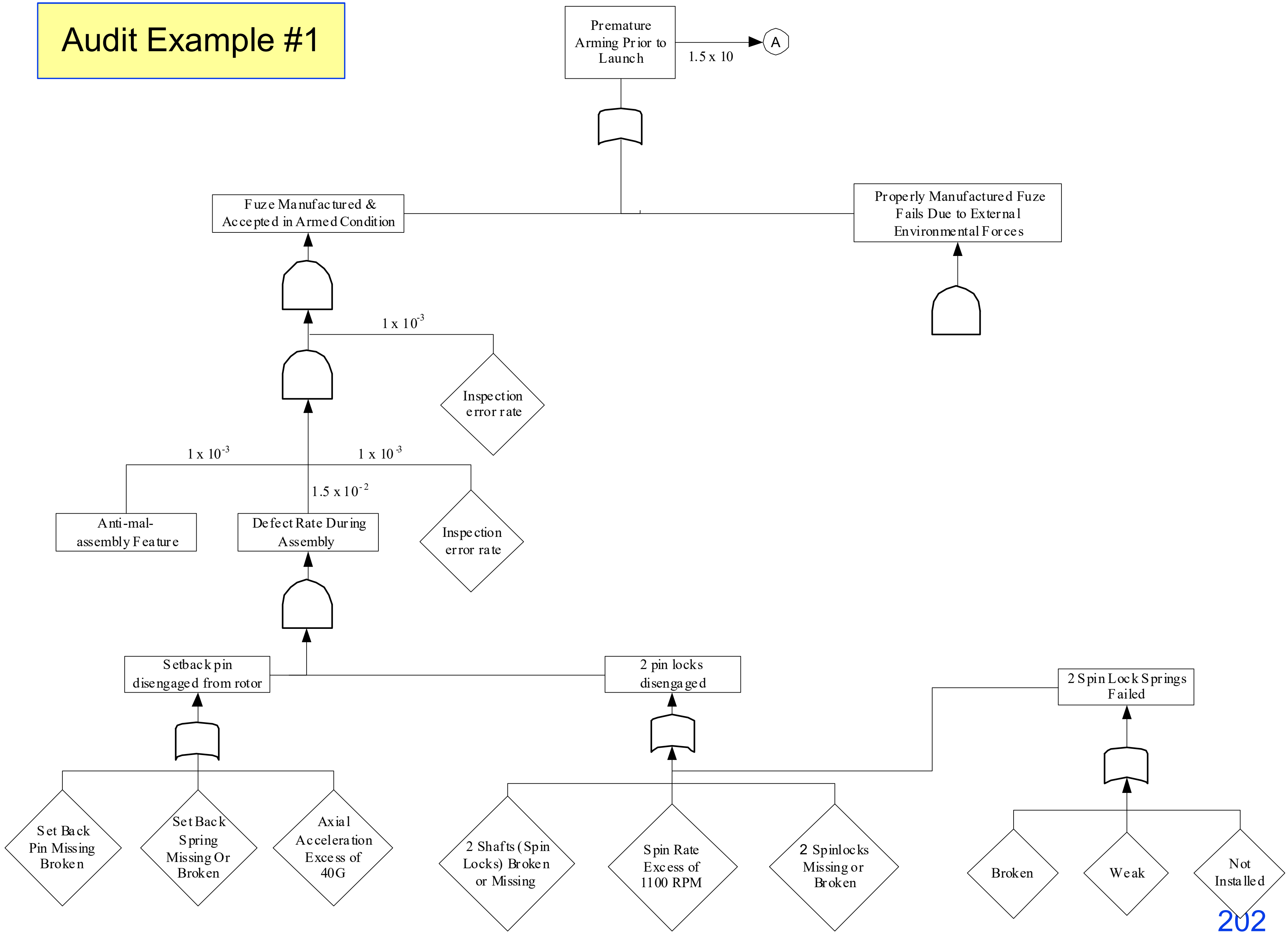
- Violation of common FT rules
  - Gate to gate
  - Multiple outputs (double connects)
  - No text in boxes
  - Inputs/outputs on side of box (vice top/bottom)
  - Incorrect symbol usage

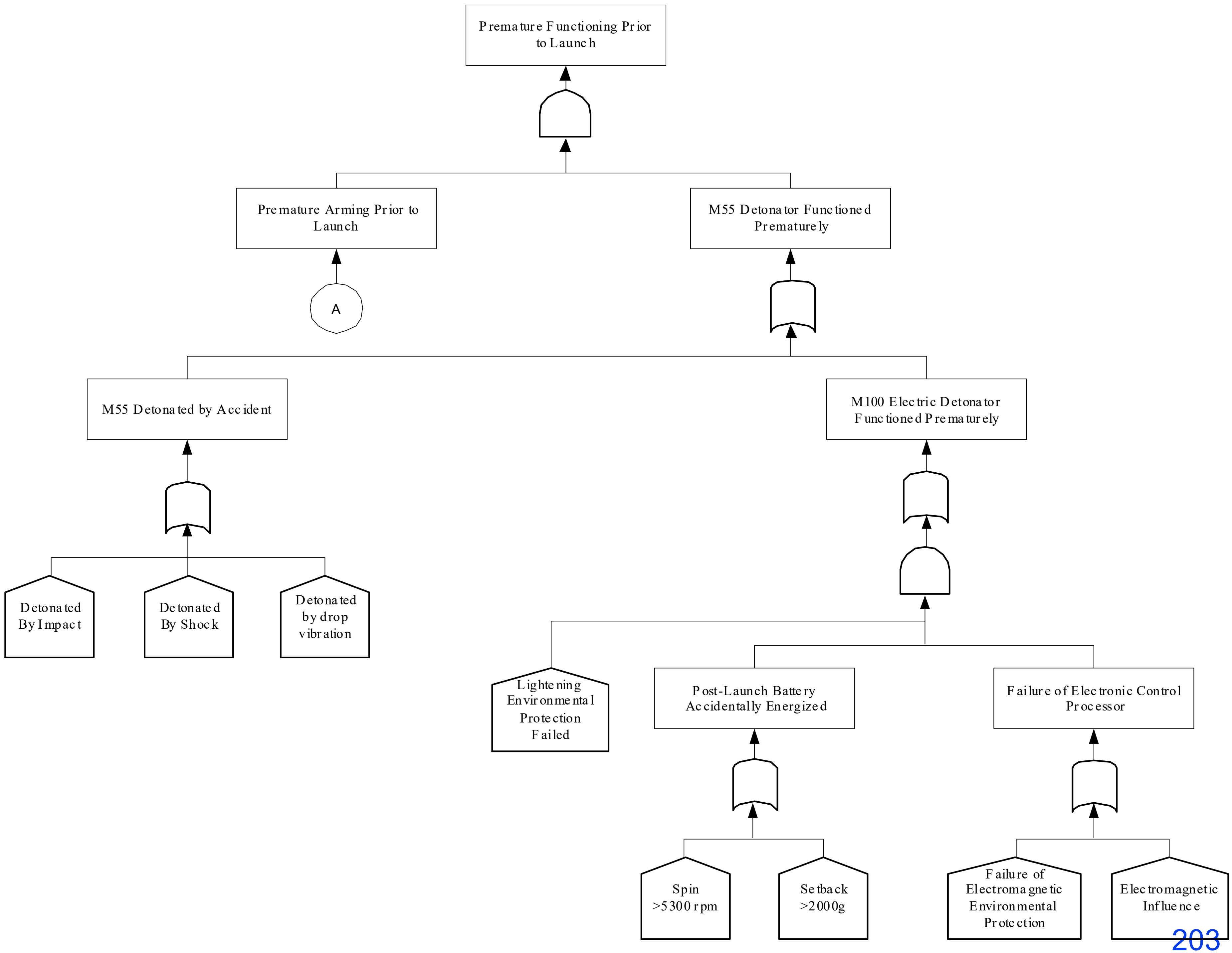
- FT Sloppiness
  - Messy diagram
  - Unreadable text (hand drawn)
  - Too small to read

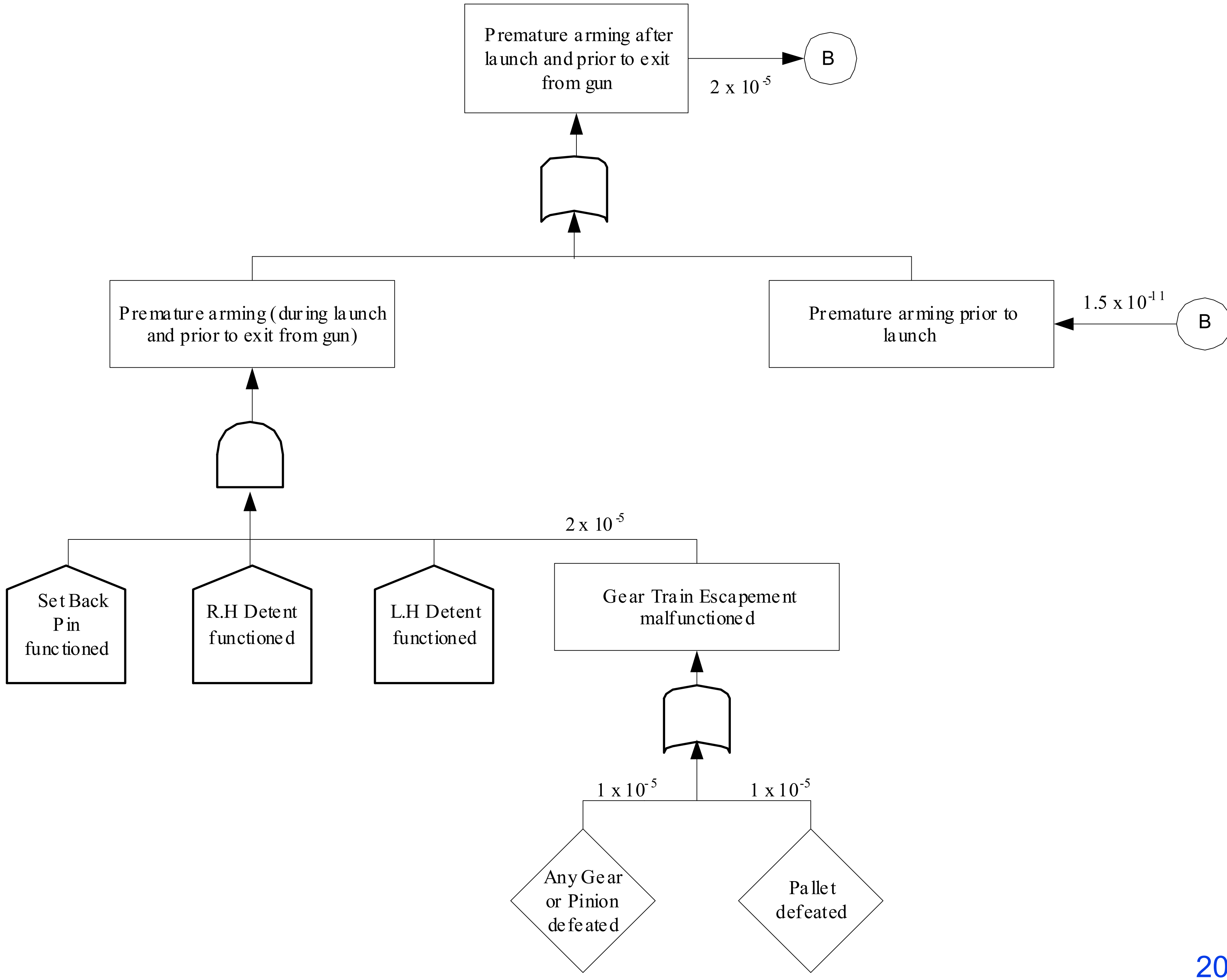
Audit Checklist

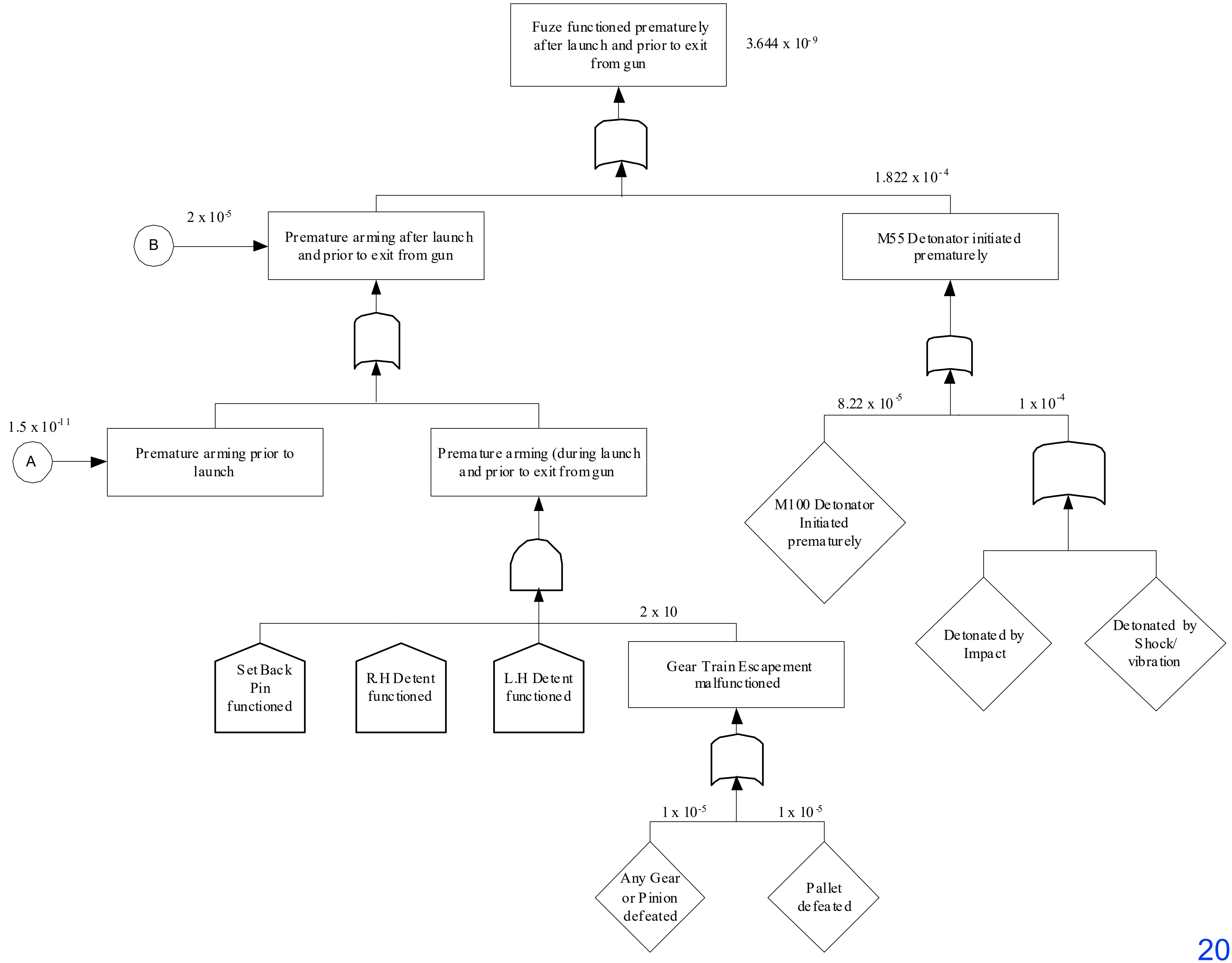
- Do CSs make sense and do they cause UE
- Are all of the CSs minimal
- Are any CSs mutually exclusive
- Is the Probability reasonable (based on data and experience)
- Do Gates appear correct
- Is failure data reasonable
- Are MOEs and MOBs correct
- Does FT diagram follow basic rules
- Do all nodes 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

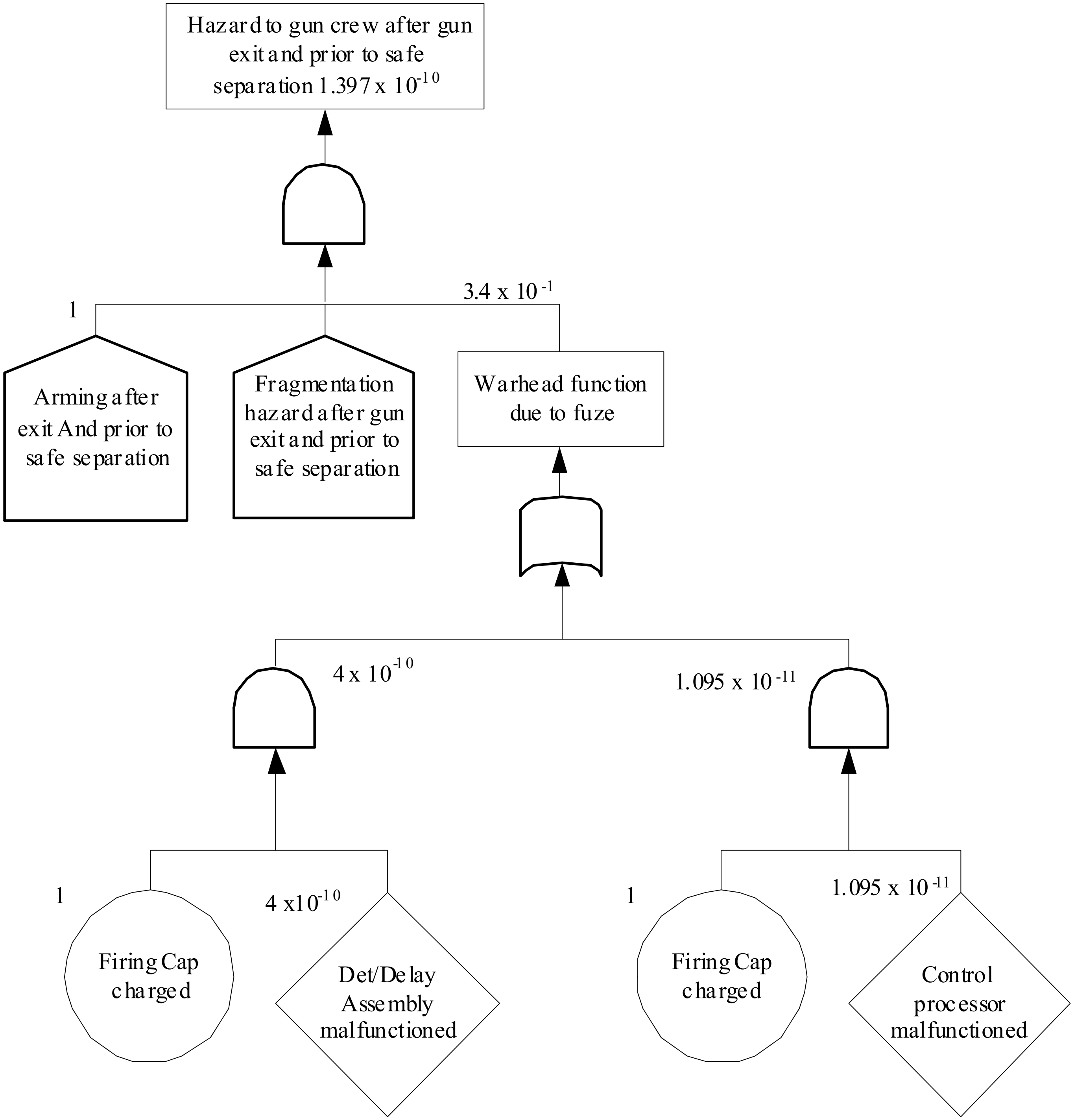



Premature Functioning Prior to Launch
Premature Arming Prior to Launch
M55 Detonator Functioned Prematurely
A
M55 Detonated by Accident
M100 Electric Detonator Functioned Prematurely
Detonated By Impact
Detonated By Shock
Detonated by drop vibration
Lightening Environmental Protection Failed
Post-Launch Battery Accidentally Energized
Failure of Electronic Control Processor
Spin >5300 rpm
Setback >2000g
Failure of Electromagnetic Environmental Protection
Electromagnetic Influence

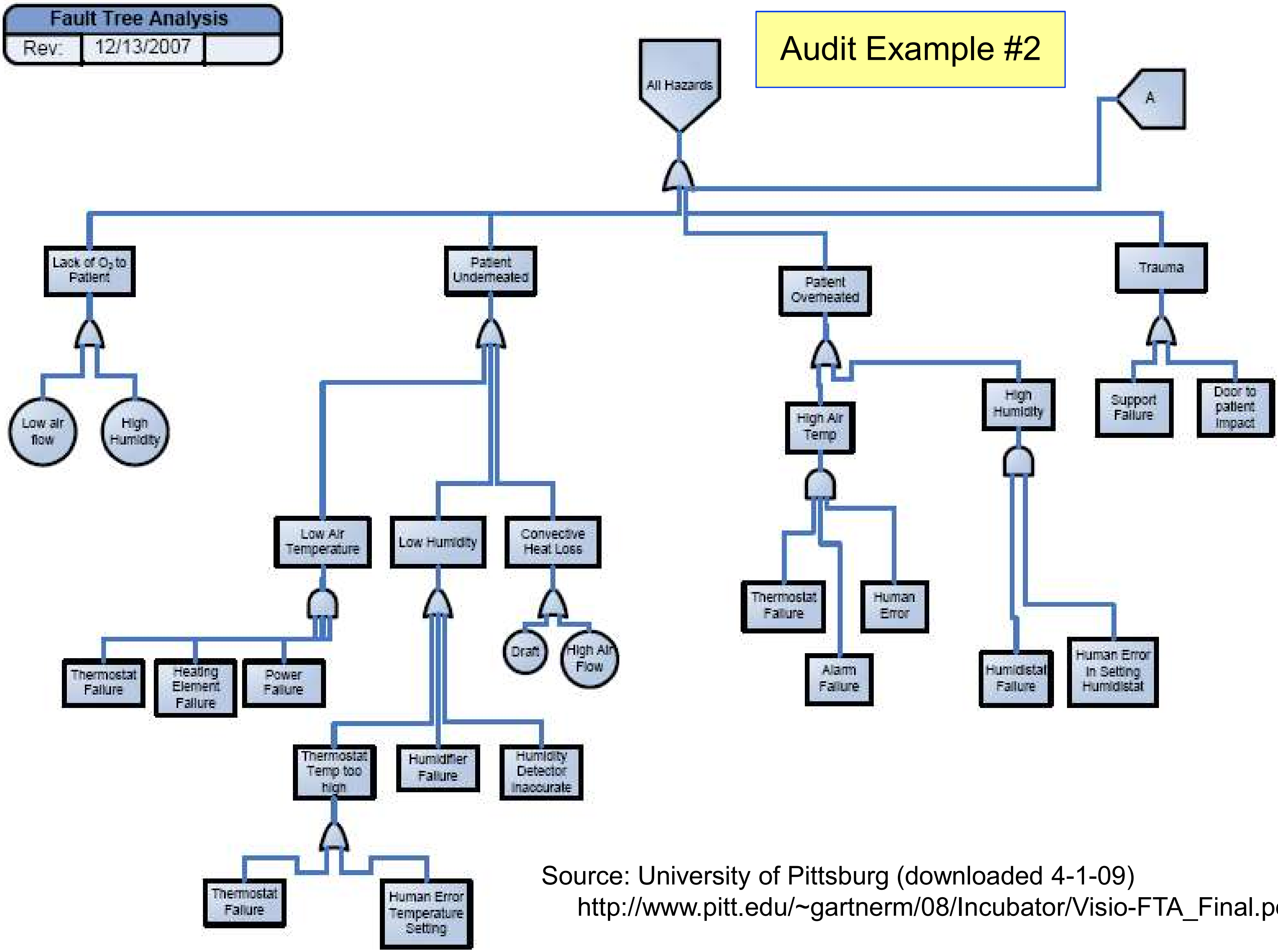

Premature arming after launch and prior to exit from gun
B
2 x 10-5
Premature arming (during launch and prior to exit from gun)
Premature arming prior to launch
1.5 x 10-11
B
2 x 10-5
Set Back Pin functioned
R.H Detent functioned
L.H Detent functioned
Gear Train Escapement malfunctioned
1 x 10-5
1 x 10-5
Any Gear or Pinion defeated
Pallet defeated



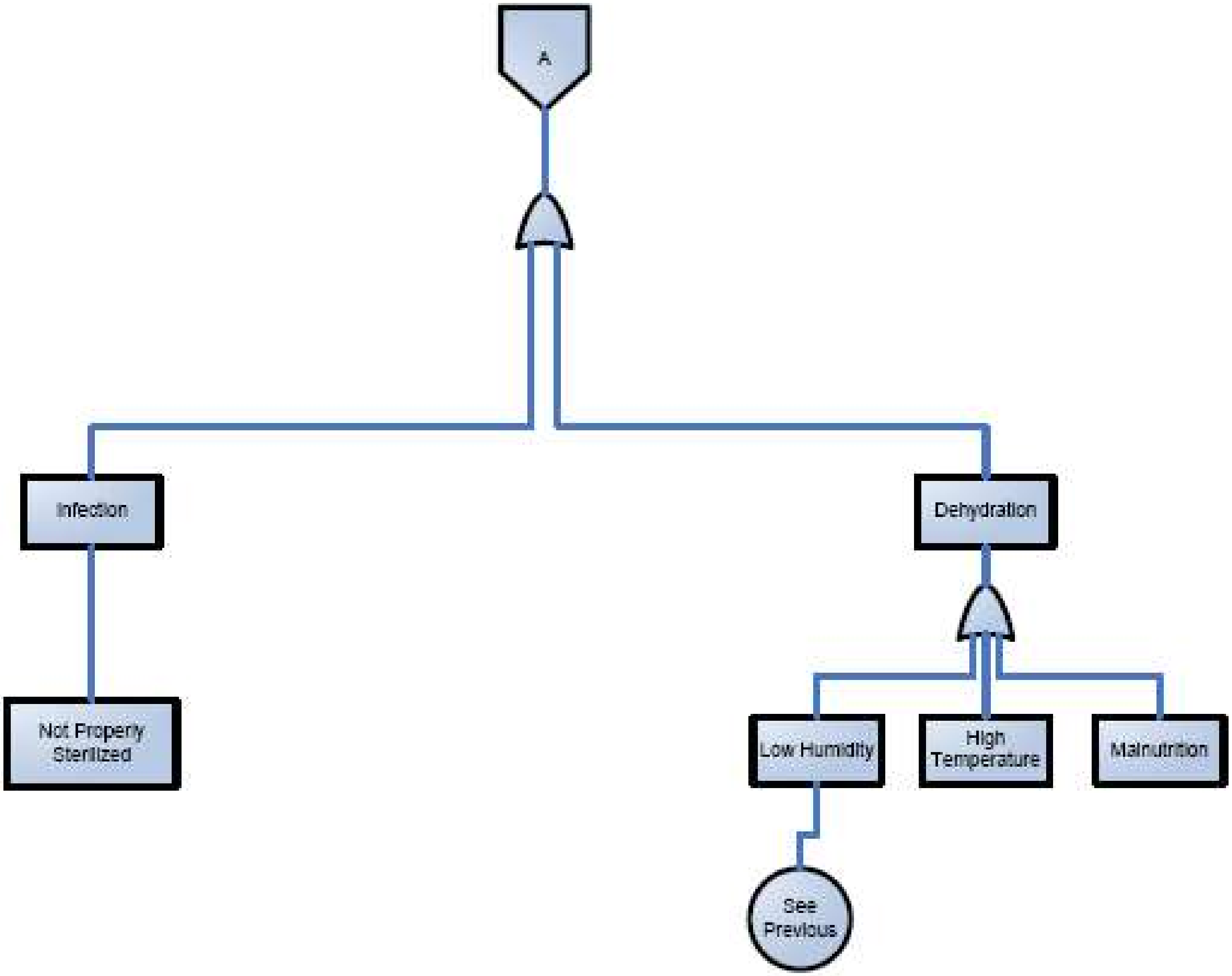
Fuze functioned prematurely after launch and prior to exit from gun
3.644 x 10-9
1.822 x 10-4
2 x 10-5
B
Premature arming after launch and prior to exit from gun
M55 Detonator initiated prematurely
8.22 x 10-5
1 x 10-4
1.5 x 10-11
A
Premature arming prior to launch
Premature arming (during launch and prior to exit from gun
M100 Detonator Initiated prematurely
Detonated by Impact
Detonated by Shock/ vibration
2 x 10
Set Back Pin functioned
R.H Detent functioned
L.H Detent functioned
Gear Train Escapement malfunctioned
1 x 10-5
1 x 10-5
Any Gear or Pinion defeated
Pallet defeated


Hazard to gun crew after gun exit and prior to safe separation 1.397 x 10-10
1
3.4 x 10-1
Arming after exit And prior to safe separation
Fragmentation hazard after gun exit and prior to safe separation
Warhead function due to fuze
4 x 10-10
1.095 x 10-11
1
4 x10-10
1
1.095 x 10-11
Firing Cap charged
Det/Delay Assembly malfunctioned
Firing Cap charged
Control processor malfunctioned

Audit Example #2



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


A
Infection
Dehydration
Not Properly Sterilized
Low Humidity
High Temperature
Malnutrition
See Previous

Audit Example #3

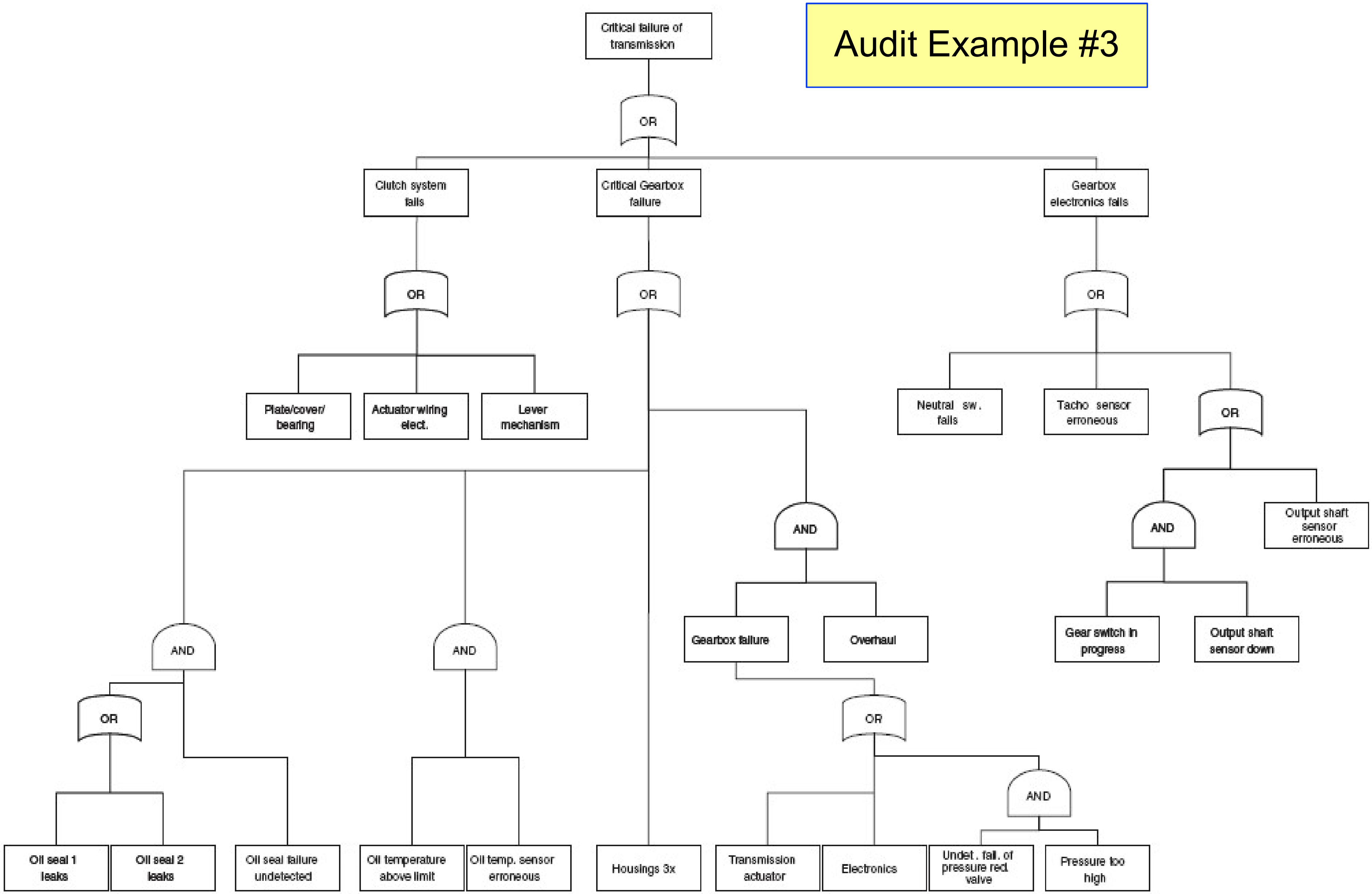


Fig. 2 Early design stage fault tree of an automatic transmission

Sources: Modelling Uncertainty In Fault Tree Analyses Using Evidence Theory, Proc. IMechE Vol. 222 Part O: Journal Risk and Reliability, 2008.

Audit Guidance

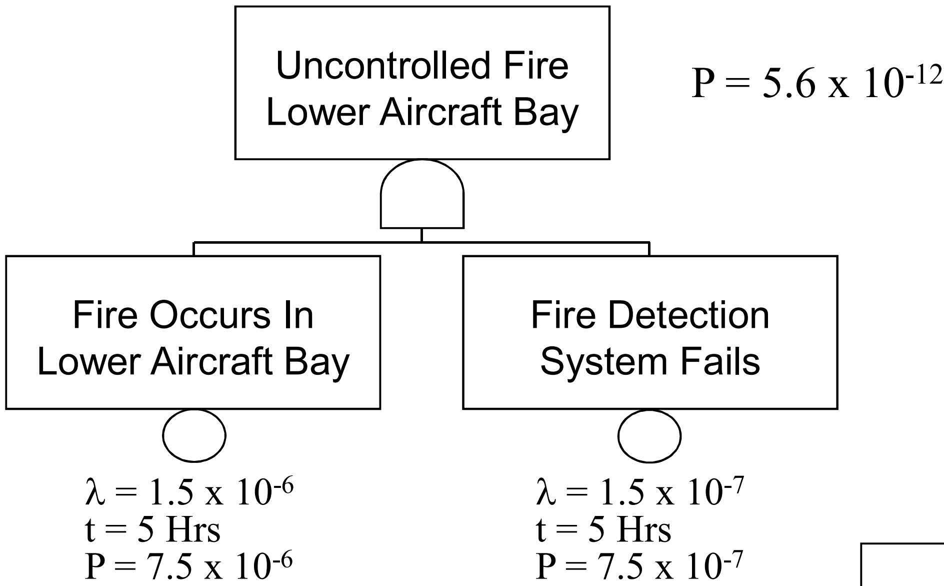
- If you have received a FTA from a contractor or supplier, it's important to obtain an independent audit of the analysis
  - Ensure the probabilities you are basing decisions on are correct
  - Require a written audit report
- Check for three defect categories
  - High, Medium and Low
- Any defects in the High category mean the FT is incorrect

--- Misc FTA Aspects ---

Latency

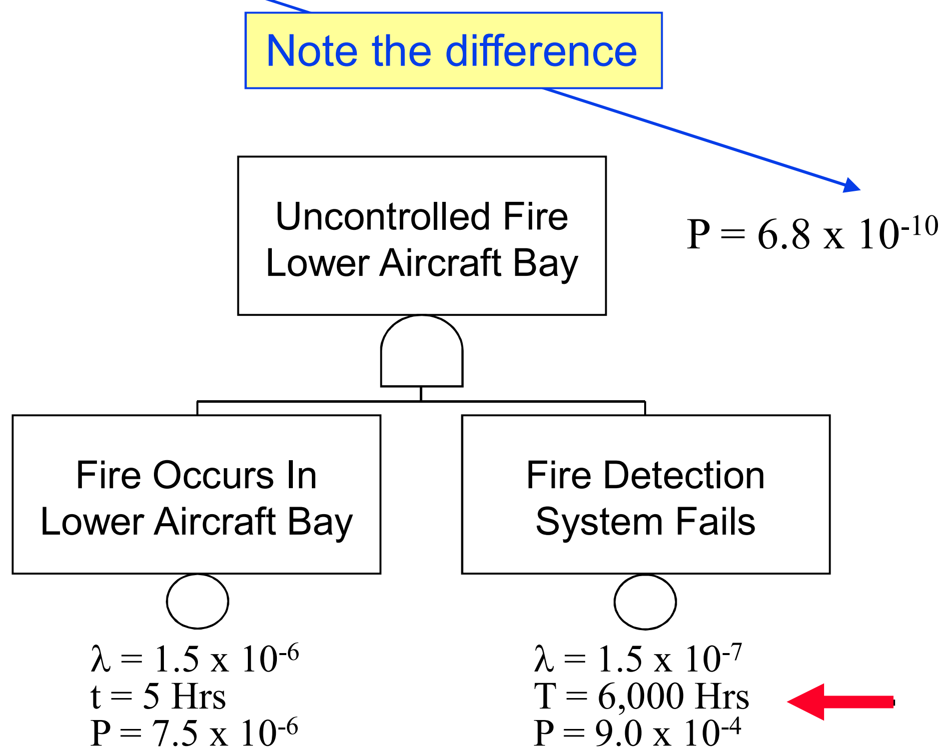
- Latency refers to a latent component failure, which is a component could be failed for some time without knowledge.
- A Latent component is a component that is not checked for operability before the start of a mission. Thus, it could already be failed at the start of the mission.
- This effectively 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



No latency

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



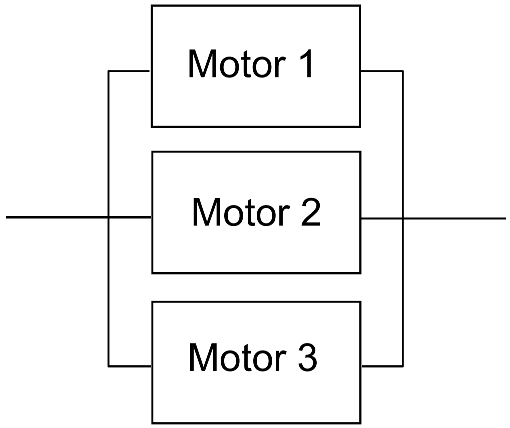
With latency

This FT assumes the Fire Detection System cannot be checked for failed state prior to flight.

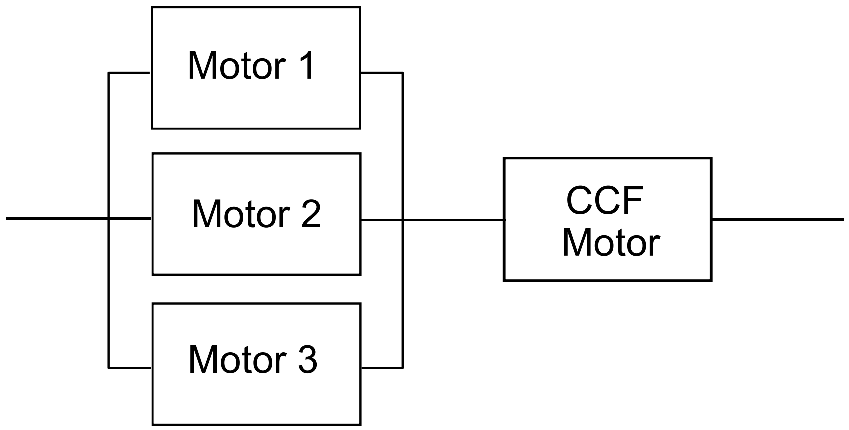
CCF

- 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

CCF

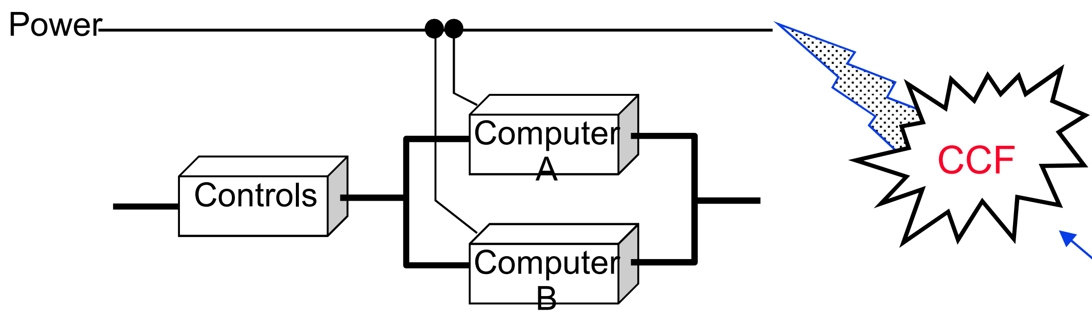


Design Intent

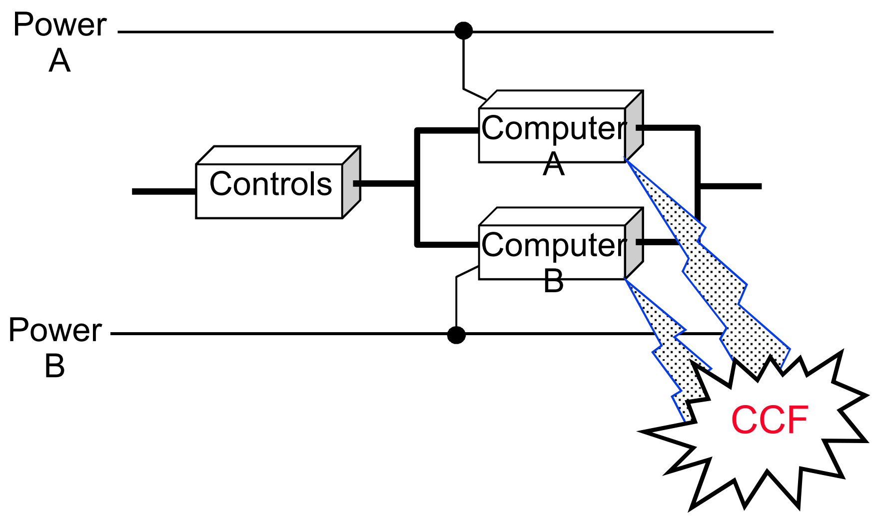


Actual Design

CCF Examples

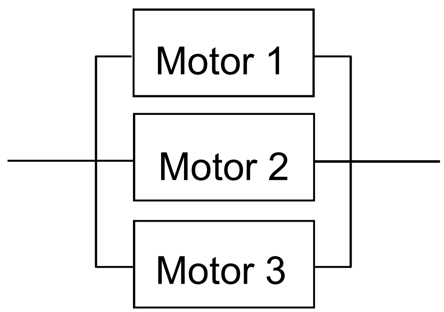


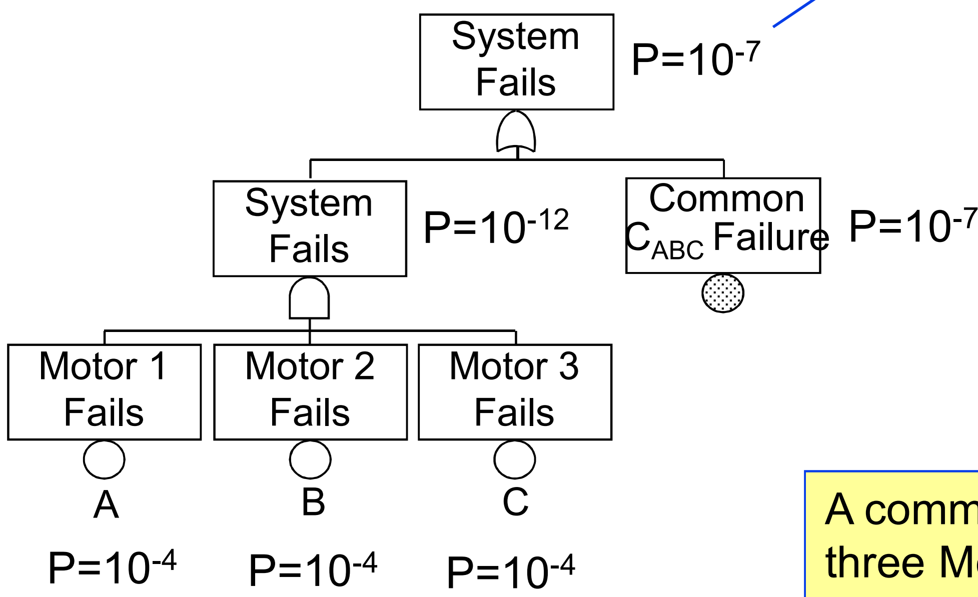
CCF Source:
- SPF

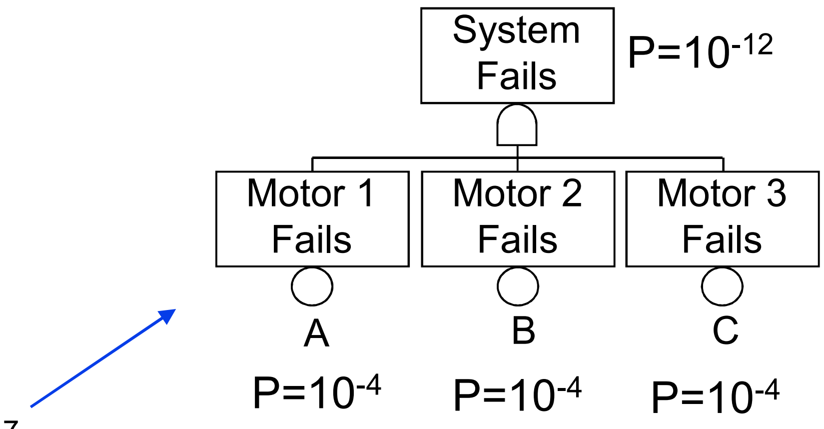


CCF Source:
- RF energy
- Flooding
- Struck by flying engine part

Example CCF FTA

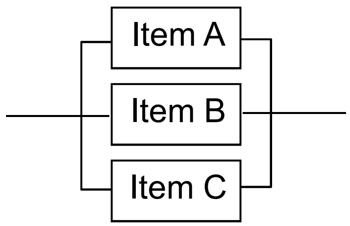


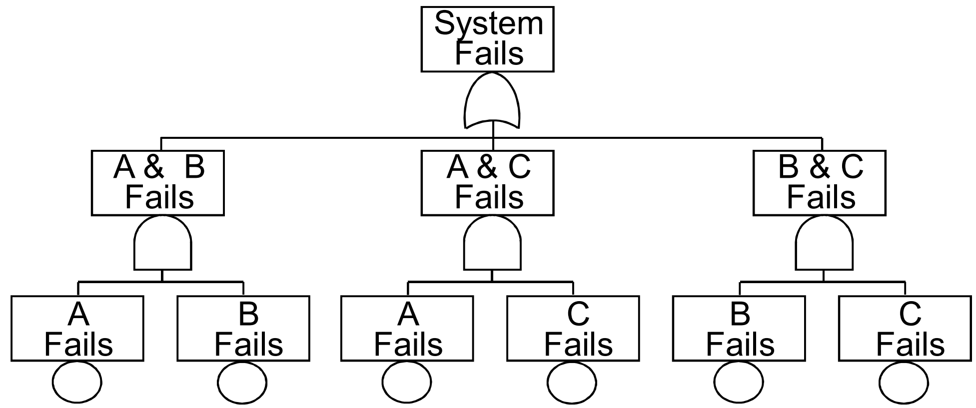


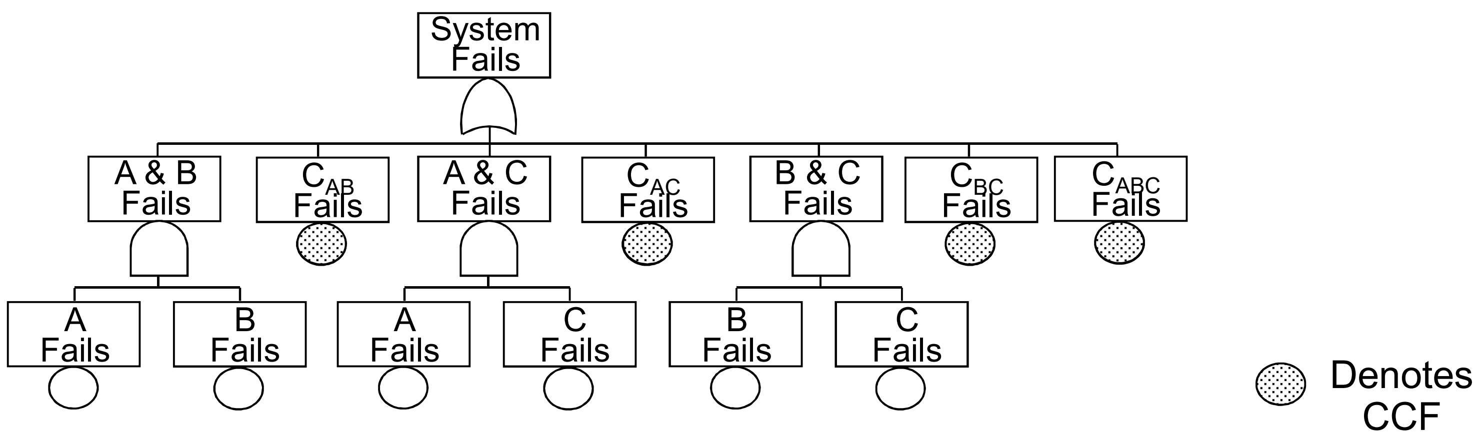


A common cause failure can cause all three Motors to fail simultaneously. This bypasses the designed redundancy. The probability of P_{ABC} is the critical factor.

Example CCF FTA

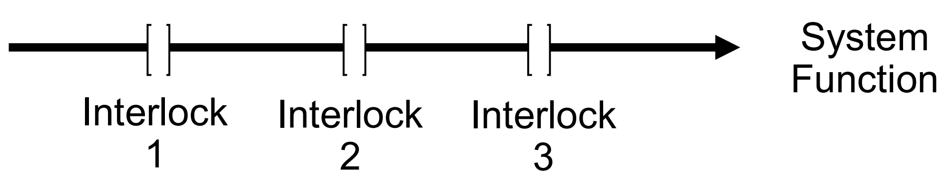




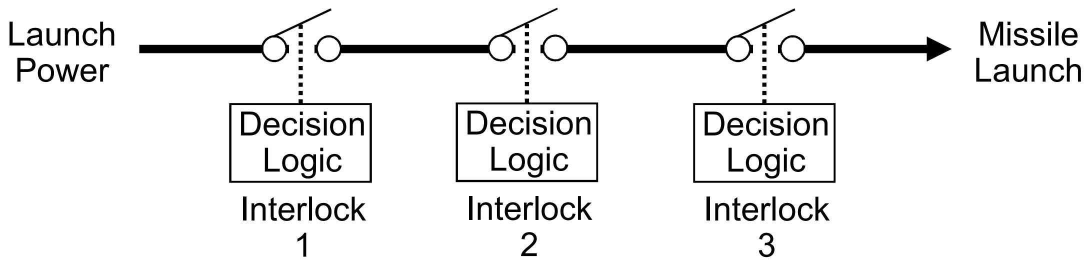


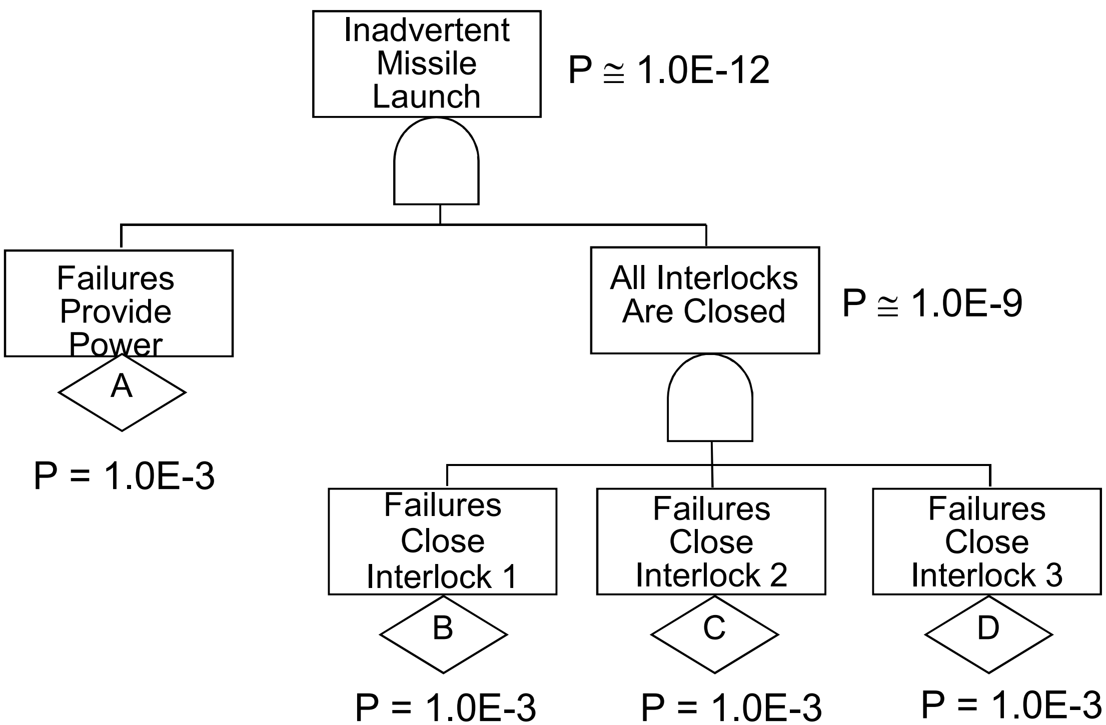
Interlocks

- 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



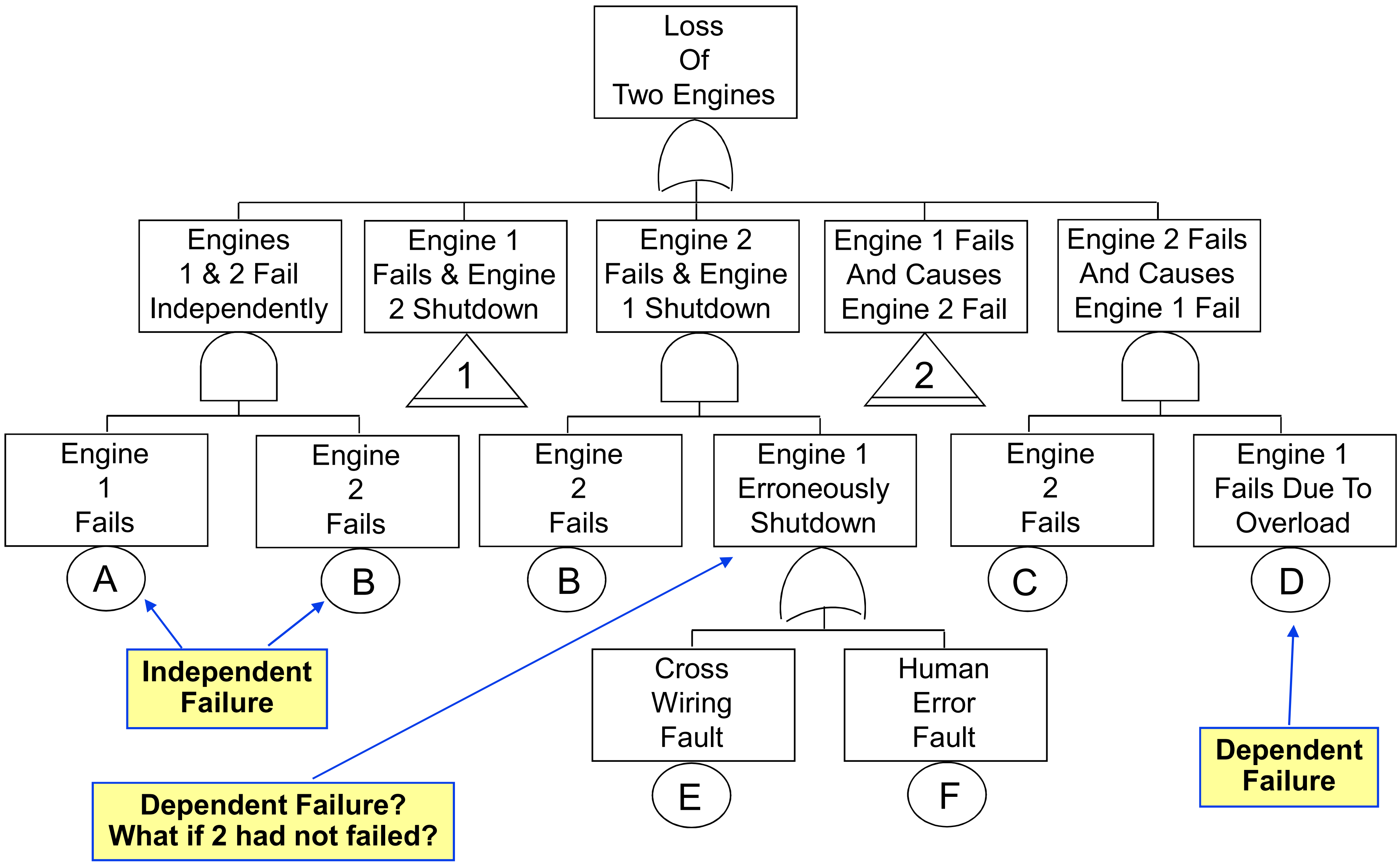


Number of Interlocks	Top Probability
0	P ≅ 1E-3
1	P ≅ 1E-6
2	P ≅ 1E-9
3	P ≅ 1E-12

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

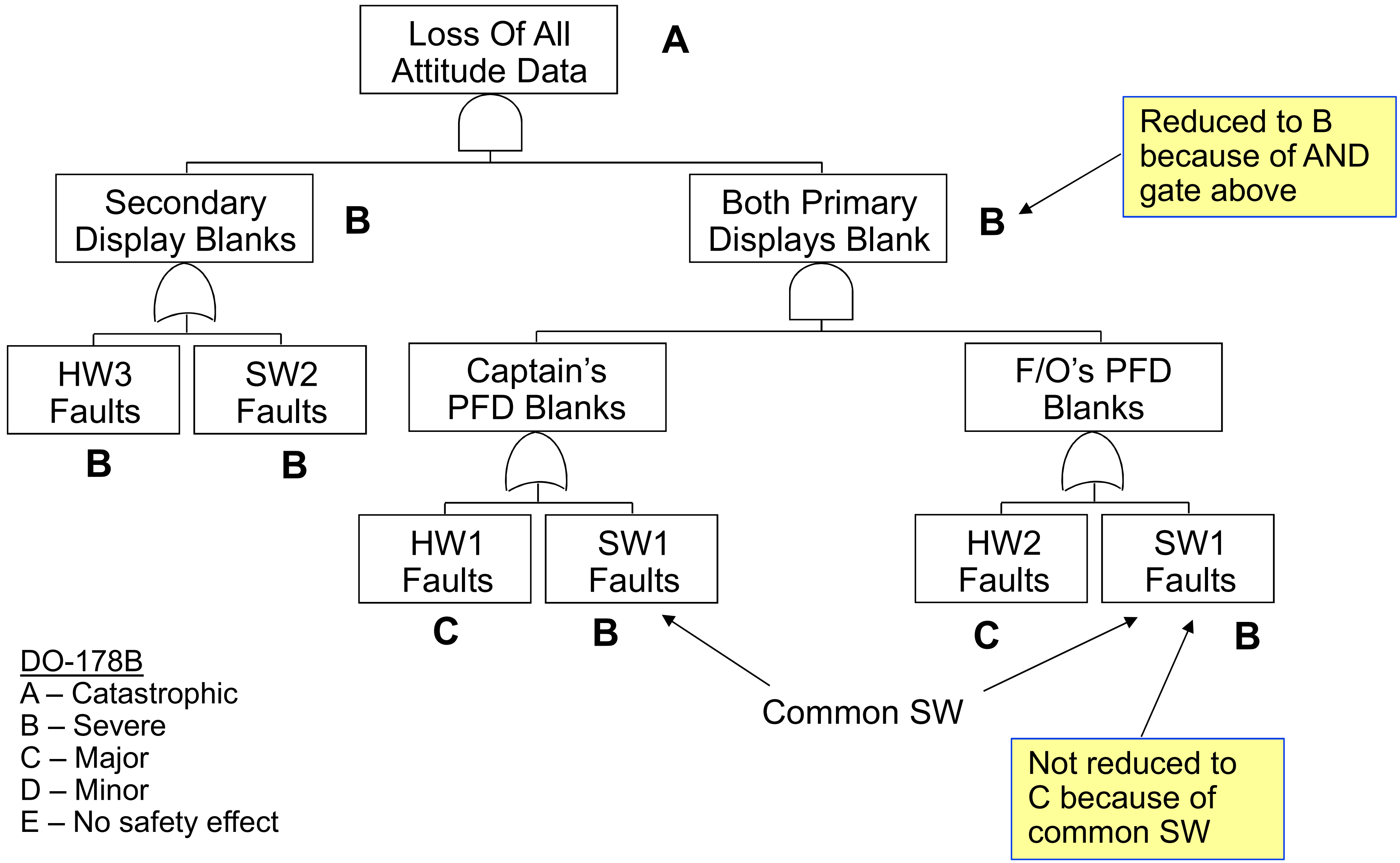


RTCA DO-178B

- 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.

Level	Definition
A	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
B	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
C	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
D	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
E	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

FT for SIL Level



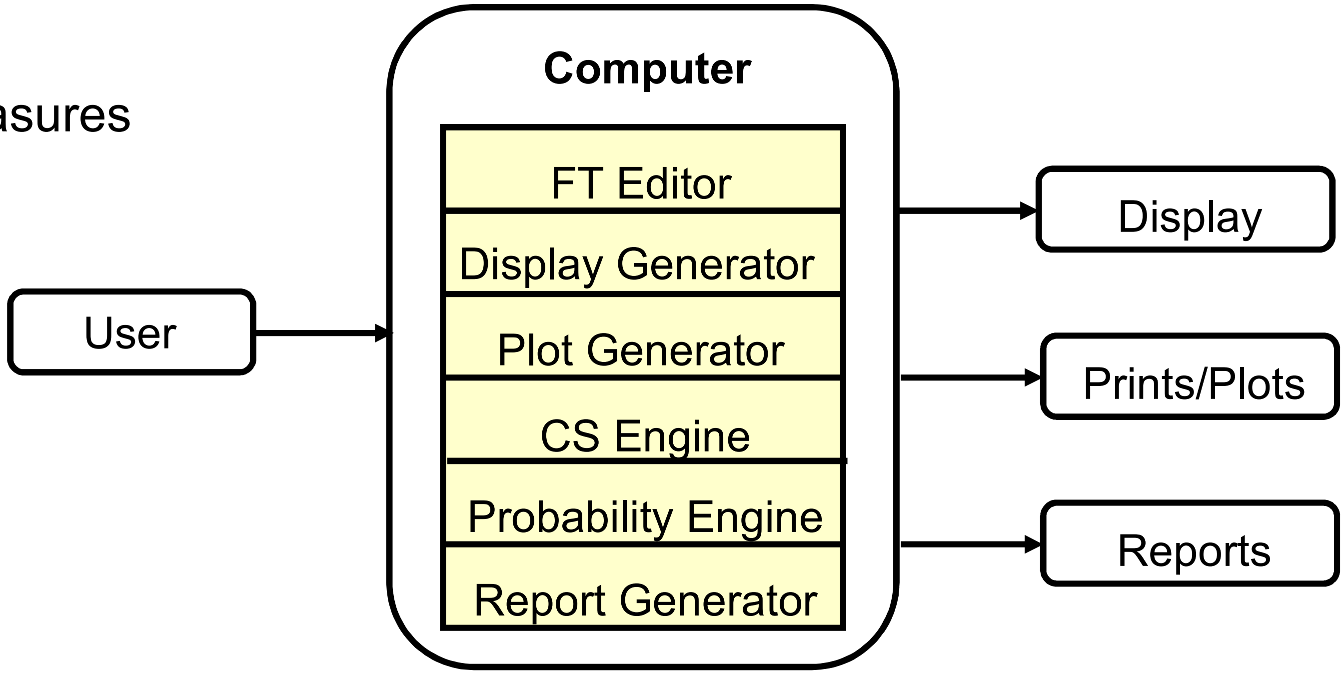
DO-178B
A – Catastrophic
B – Severe
C – Major
D – Minor
E – No safety effect

PFD=Primary Flight Display

--- FTA Software ---

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

Basic Tasks of FTA Software



Properties of FT Software

	Required	Desired
• Construction (create, edit, copy, past)	X	
• Generate Min CS	X	
• Generate probabilities (top, gates)	X	
• Employ CS cutoff methods (order, probability)		X
• Numerical accuracy	X	
• Generate reports	X	
• Detect tree logic loops		X
• Data export capability (tree structure, failure data)		X
• Graphic export capability (BMP, JPG)		X
• Tree Pagination for prints	X	
• User select print size		X
• Program fit on a floppy disk		X
• Notes on FT print/plot		X
• Find feature		X
• Undo feature		X

Properties (continued)

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

Commercial FT Software

CAFTA	CAFTA sales 650-384-5693 (part of EPRI)
Fault Tree+	www.Isograph.com
Item	www.itemsoft.com
Relex	www.relexsoftware.com
FaulTrEase	http://www.chempute.com/faultrea.htm
Risk Spectrum FT	http://www.relcon.com
Shade Tree	http://www.qrainc.com
Tree-Master	http://www.mgtsciences.com
FTA Pro	http://www.dyadem.com
OpenFTA (free source)	http://www.openfta.com
Logan	http://www.arevarmc.com/logan-faulttreeanalysis.php
RAM Commander	http://www.aldservice.com
Saphire (free?)	https://saphire.inl.gov/faq.cfm

Purchase Considerations

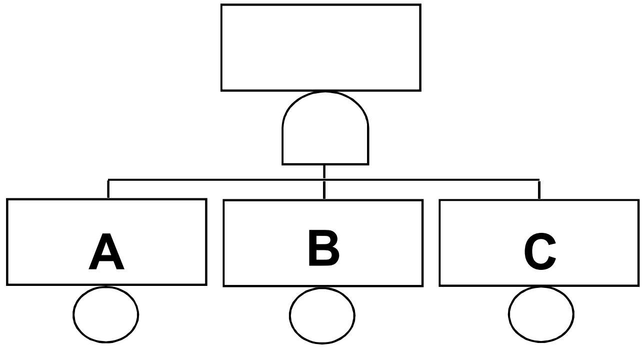
- Many commercial FT codes are available -- some good and some ?
- Know FT tool limits and capabilities
  - Tree size, Cut set size, Print size, numerical accuracy
  - Easy to use (without tech support)
  - Single phase, multi-phase
- Understand algorithms
  - Some codes have errors in approximations 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)



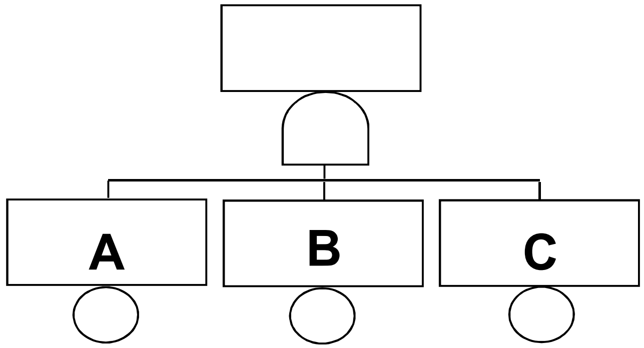
Incorrect	Correct
1 Hr Mission $P_{TOP} = P_A \bullet P_B \bullet P_C$ $= \lambda_A T \bullet \lambda_B T \bullet \lambda_C T$ $= (1x10^{-6})(1) \bullet (1x10^{-6})(1) \bullet (1x10^{-6})(1)$ $= 1x10^{-18}$ ***Adjusted for 100 Hr Mission*** $P_{TOP} = P_{TOP}$ x100 Hrs $= (1x10^{-18})(100)$ $= 1x10^{-16}$	***100 Hr Mission*** $P_{TOP} = P_A \bullet P_B \bullet P_C$ $= \lambda_A T \bullet \lambda_B T \bullet \lambda_C T$ $= (1x10^{-6})(100) \bullet (1x10^{-6})(100) \bullet (1x10^{-6})(100)$ $= 1x10^{-12}$

Mission Calculation Errors (cont'd)

Example 2

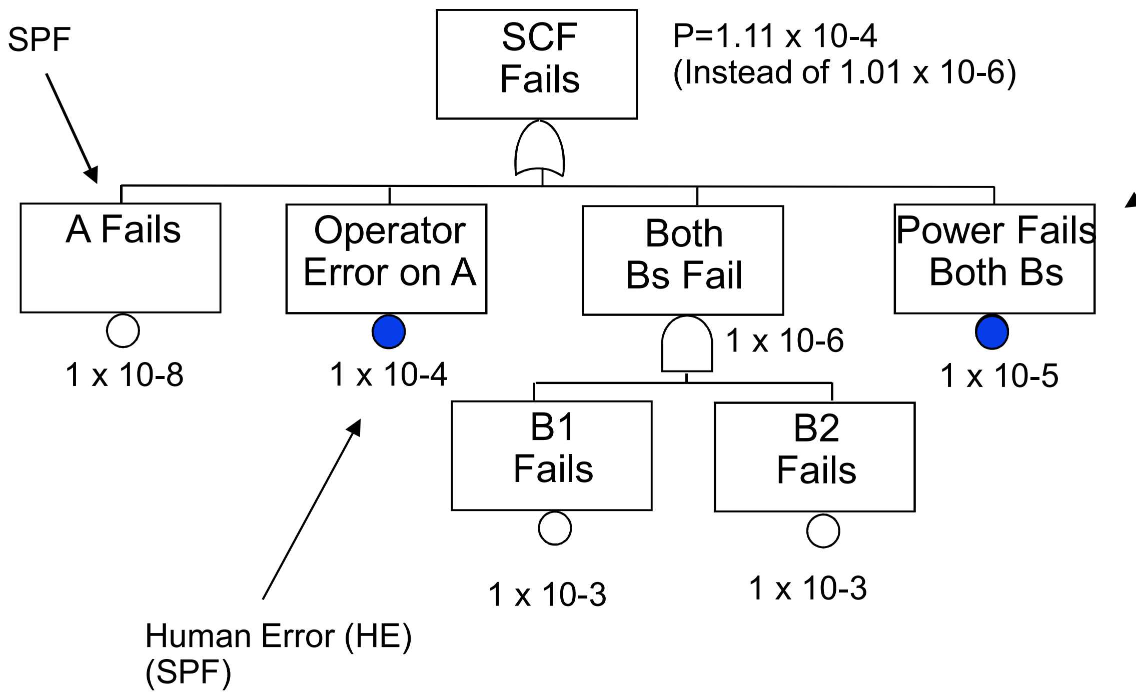
- *Shortcut error (multiply λ's)*

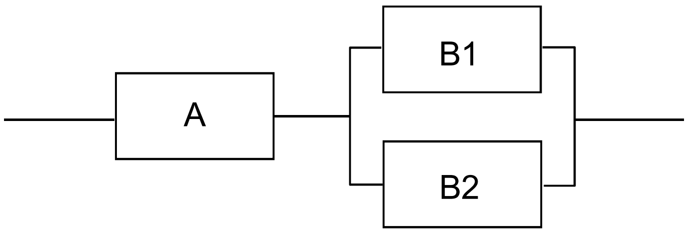
$P \neq (\lambda_{TOTAL}) \times (\text{Mission Time})$



Incorrect	Correct
100 Hr Mission	***100 Hr Mission***
$P_{TOP} = \lambda_{TOP} \bullet T$	$P_{TOP} = \lambda_A T \bullet \lambda_B T \bullet \lambda_C T$
$= (\lambda_A \bullet \lambda_B \bullet \lambda_C) \bullet (T)$	$= (\lambda_A \bullet \lambda_B \bullet \lambda_C) \bullet (T^3)$
$= (1x10^{-6}) \bullet (1x10^{-6}) \bullet (1x10^{-6})(T)$	$= (1x10^{-6})(1x10^{-6})(1x10^{-6})(10^6)$
$= 1x10^{-18} (T)$	$= 1x10^{-12}$
$= 1x10^{-18} (100)$	
$= 1x10^{-16}$	

Ignoring Critical Items





Safety Critical Function (SCF)

Some important items are often ignored or overlooked

- Latency
- Human error
- CCFs

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