EUROPEAN ORGANISATION
FOR THE SAFETY OF AIR NAVIGATION

EUROCONTROL

FAULT TREE ANALYSIS (FTA) GUIDANCE MATERIAL

SAF.ET1.ST03.1000-FTA-01-00

Edition Number	:	1.0
Edition Date	:	01.06.05
Status	:	Released Issue
Intended for	:	General Public

EUROPEAN AIR TRAFFIC MANAGEMENT

DOCUMENT CHARACTERISTICS

TITLE		
Fault Tree Analysis (FTA) Guidance Material		
	EATMP Infocentre Reference:	
Document Identifier	**Edition Number:**	1.0
	Edition Date:	01.06.05

Abstract

Keywords

SAM	SSA
Safety Assessment	Fault Tree Analysis
ATM Procedure	Fault Tree
PSSA	

Contact Person(s)	**Tel**	**Unit**
Patrick MANA	93295	DAP/SAF

STATUS, AUDIENCE AND ACCESSIBILITY

Status		**Intended for**		**Accessible via**	
Working Draft	☐	General Public	☑	Intranet	☐
Draft	☐	EATM Stakeholders	☐	Extranet	☐
Proposed Issue	☐	Restricted Audience	☐	Internet (www.eurocontrol.int)	☐
Released Issue	☑	*Printed & electronic copies of the document can be obtained from the EATMP Infocentre (see page iii)*			

ELECTRONIC SOURCE

	H:\Private\TRS\Fault Tree\deliverables

Host System	Software	Size
Windows_XP	Word	xxx Kb

DOCUMENT APPROVAL

The following table identifies all management authorities who have successively approved the present issue of this document.

AUTHORITY	NAME AND SIGNATURE	DATE
Please make sure that the EATMP Infocentre Reference is present on page ii.		
Chairman of the Safety Assessment Methodology Task Force	P.MANA	01.06.05

DOCUMENT CHANGE RECORD

The following table records the complete history of the successive editions of the present document.

EDITION NUMBER	EDITION DATE	INFOCENTRE REFERENCE	REASON FOR CHANGE	PAGES AFFECTED
0.1	05/12/2003		Creation of the working draft	All
0.2	19/01/2004		Comments / Evolution of the document	All
0.3	13/02/2004		Comments / Evolution of the document	All
1.0	01.06.05		Comments / Evolution of the document	All

CONTENTS

Acronyms

N	None
FHA	Functional Hazard Assessment
FTA	Fault Tree Analysis
PSSA	Preliminary System Safety Assessment
SAM	Safety Assessment Methodology
SO	Safety Objective
SR	Safety Requirement
SRS	Safety Requirement Specification
SSA	System Safety Assessment

Documents

Ref. 1	Guidelines for the Safety Assessment of ATM Procedures (SAAP) EUROCONTROL Ref: SAF.ET1.ST03.1000-SAAP-01-00
Ref. 2	Reliability, availability, maintainability and safety assessment John Wiley and Sons, Inc., 1991 A. Villemeur
Ref. 3	Safety and Performance Requirements Standard For Initial Air Traffic Data Link Services In Continental Airspace (ED120) RTCA SC-189 / EUROCAE WG-53
Ref. 4	Guidance Material A of PSSA – Chapter 3 EUROCONTROL Ref: SAF.ET1.ST03.1000-MAN-02-03-A

Executive summary

This document provides guidance to apply Fault Tree Analysis in the framework of the Safety Assessment Methodology (SAM). Therefore, this document constitutes a SAM Level 2 Guidance.

Specifically, it introduces the basic principles underlying this very widely spread and used technique in safety assessment of many domains (not only aviation). It details the advantages and limitations of such technique when undergoing Safety Assessment of Air Navigation System.

It explains how to specifically use Fault Tree Analysis for both PSSA (Preliminary System Safety Assessment) and SSA (System Safety Assessment):

- in the PSSA as a technique supporting the Top-Down apportionment (or budgeting) of Safety Objectives (acceptable frequency of occurrence of hazard) into Safety Requirements to the elements of the architecture that cause or contribute to this hazard occurrence.

- In the SSA as a technique mainly supporting the Bottom-Up verification of Safety Objective satisfaction.

1

1 - INTRODUCTION

1.1. Relation to SAM

Guidance Material

This Fault Tree Analysis (FTA) Guidance Material should be perceived as a part of EUROCONTROL Safety Assessment Methodology (SAM).

The SAM is made of three levels of material (See Figure 1.1).

- Level1: The <u>methodology</u> following logically the steps:
    - FHA (Functional Hazard Assessment) (SAF.ET1.ST03.1000-MAN-01),
    - PSSA (Preliminary System Safety Assessment) (SAF.ET1.ST03.1000-MAN-02),
    - SSA (System Safety Assessment) (SAF.ET1.ST03.1000-MAN-03)
- Level 2:There are two types of <u>Guidance Material</u> :
    - Providing further detailed information on the use of various techniques to achieve some parts of FHA or PSSA or SSA;
    - Addressing specific system element throughout all the methodology steps such as ATM procedure or Software.
- Level 3: <u>Examples</u> of application of various techniques to real safety assessment.

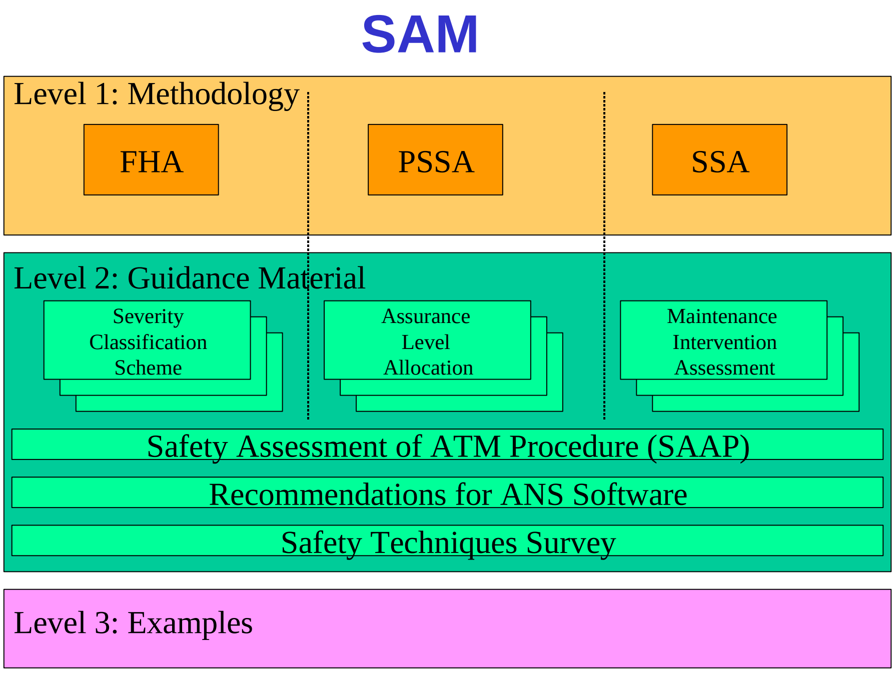

SAM
Level 1: Methodology
FHA
PSSA
SSA
Level 2: Guidance Material
Severity Classification Scheme
Assurance Level Allocation
Maintenance Intervention Assessment
Safety Assessment of ATM Procedure (SAAP)
Recommendations for ANS Software
Safety Techniques Survey
Level 3: Examples

Figure 1.1: SAM structure

1.2. Purpose

This document constitutes a Level 2 document of the SAM (Part IV Annex K).

This document does not prescribe the only way of performing Fault Tree Analysis. It rather provides one possible way to achieve such a task.

The purpose of these guidelines is to provide more insights for the Fault Tree Analysis (FTA) by:

- Providing Guidance Material to the SAM,
- Providing the link with the SAM PSSA and SSA process,
- Providing practical advices on the use of FTA technique.

1.3. Structure of this document

This document is structured in two main parts;

- **Main body (Chapters)**, which describe how and where to use FTA as part of a safety assessment;
- **Appendixes**, which provide background material.

1.4. Target audience

This document is specifically targeted at:

Safety practitioners: Conduct and document the safety assessment processes.

They are responsible for:

- the performance and documentation of FHA, PSSA and SSA processes,
- the link between the programme/project and the safety assessment process,
- the methodological support to the different steps of the safety assessment process.

Safety Reviewers: Review for Verification and Validation of the safety documentation

They could be system integrators, regulators.

They are responsible for the verification and review of the safety documentation.

Safety managers: Responsible for the safety management of a Project/Programme, they are in charge with the following.

- specification of a Project/Programme Safety Plan in order to achieve the applicable safety objectives.

- gathering Project/Programme inputs for System Safety Assessment Documentation.

- allocating the responsibilities for the preparation, verification and review of the system safety documentation.

- review/comment/approval of FHA, PSSA, SSA analysis in accordance with the project/Programme Safety Plan.

1.5. Readership

The following table suggests a minimum reader's attention to this document.

	Safety Manager	Safety Practitioner		Safety Reviewer	Other roles (System designer, ..)
		Beginner	Expert		
Chapter 1 – Introduction	✓	📖	✓	✓	✓
Chapter 2 – Basic Principles of FTA	✓	📖	✓	✓	✓
Chapter 3 – Fault Tree Description	✓	📖	✓	✓	✓
Chapter 4 – FTA in support of the PSSA	✓	📖	📖	📖	✓
Chapter 5 – FTA in support of the SSA	✓	📖	📖	📖	✓
Appendix A – Standardized basic symbols	N/A	📖	✓	✓	N/A
Appendix B – Fault Tree Construction Fundamentals	N/A	📖	✓	✓	N/A
Appendix C – Boolean algebra / Minimal Cut Set / Qualitative Approach	N/A	📖	📖	📖	N/A
Appendix D – Quantitative Approach	N/A	📖	📖	📖	N/A
Appendix E – Quantitative allocation through the fault tree	N/A	📖	📖	📖	N/A

📖: Detailed knowledge;

✓ : Aware;

N/A: Not Applicable.

1.6. When is the Fault Tree Analysis used?

A Fault Tree Analysis is recommended for any:

- investigation to define incident/accident causes,
- allocation of safety requirements,
- verification that safety objective(s) are met or not.

1.7. What is the Fault Tree Analysis (FTA)?

Fault Tree Analysis is concerned with the identification and analysis of conditions and factors which cause or contribute to the occurrence of a defined undesirable event, usually one which significantly affects system safety, performance, economy, or other required characteristics. FTA is intensively applied to the systems safety assessment.

1.8. What about software tools?

Software tools are used to:

- edit fault trees,
- determine minimal cut sets in support of qualitative evaluation,
- perform quantitative evaluation,
- import/export data (from/to reliability data bases, or to report results).

1.9. Safety / Dependability

It is to be noted that this guidance material is closely linked to the Safety Assessment Methodology (perceived as a part of it), thus the guidelines, explanations and advice appearing in this document are Safety oriented.

Moreover, it is to be highlighted that, in this guidance material, quantitative Safety Objectives result, through allocation process, into Safety Requirements addressing Dependability: reliability, availability, integrity, maintainability.

2

2 - BASIC PRINCIPLES OF FTA

This Guidance Material describes Fault Tree Analysis technique (FTA), and gives guidance on its application in the frame of the Safety Assessment Methodology (SAM).

FTA History

Historically, FTA appeared in the beginning of the 60's to assess and improve the reliability of a missile launch system. It is frequently used since the middle of the 60's in many industrial domains as aeronautics, chemical industry, nuclear industry, ground transportation (railway, automotive) etc.

A deductive method

Fault tree analysis is basically a deductive (top-down) method of analysis aimed at pinpointing the causes or combinations of causes that can lead to the defined undesired top event. The question to be answered is "why does the undesired event happen?". The analysis aims at building a Fault Tree model that allows:

- to perform a Qualitative evaluation of the top event occurrence (causes and their combination),
- and furthermore, as far as data (MTBF, MTTR, λ, μ, probability, etc.) are input to the causes, to perform a Quantitative assessment (probabilistic evaluation) as well.

Why perform an FTA?

There are several reasons for performing a fault tree analysis independently of, or in conjunction with, other safety analyses. These include:

- the identification of the causes and combinations of causes leading to the top event, for example hardware equipment failure modes, operator mistakes, environmental adverse conditions, software faults, inappropriate procedures, functional failures,
- the determination of whether a particular system safety measure meets a stated objective,
- the determination of the factor(s) which most seriously affect a particular safety measure and the changes required to improve that measure,
- the identification of elements which could cancel the benefits of specific redundancies: common events (a same malfunction affecting both the main element and its redundancy) or common cause failures (factors that invalidate the assumption of malfunctions independency, e.g. redundant similar components experiencing the same adverse environmental conditions or having been produced by the same manufacturer).

Graphical representation of causes' combination contributing to a "feared" event

The fault tree itself is an organized graphical representation of the conditions or other factors causing or contributing to the occurrence of a defined “feared” (undesirable) event, referred to as the "top event".

Particularly suited to the analysis of complex systems

The fault tree is particularly suited to the analysis of complex systems comprising several functionally related or dependent subsystems with different performance objectives. This is especially true whenever the system design requires the collaboration of many specialized technical design groups. Examples of systems to which fault tree analysis is commonly applied include nuclear power generating stations, aeroplanes, communication systems, chemical and other industrial processes, etc.

Fault tree growth reflects the progress of the design

The development of the fault tree should start early in the system design stage. The growth of the fault tree should be such that it reflects the progress of the design. Thus an increased understanding of the failure modes will be obtained as the design proceeds. The "analysis concurrent with design" allows for early systems design change as significant failure modes and their contribution to the undesired events are identified. The final fault trees will be large, in which case a specialized software tool will be needed to handle them.

It is important to note that fault tree events are not confined solely to equipment (hardware or software) failures, but include all conditions or other factors which are relevant to the top event for the system concerned, like human errors, inappropriate procedures and adverse environment conditions.

Steps to be done

In order to use the fault tree technique effectively as a method for system analysis, the procedure shall consist of at least the following steps:

- definition of the scope of the analysis,
- familiarization with the design (functions, architecture) and operation of the system,
- definition of the top event(s),
- construction of the fault tree(s),
- analysis of the fault tree logic,
- quantitative evaluation (when probabilistic data available),,
- reporting on results of the analysis.

Timing or sequencing of events

Boolean reduction is applicable in fault trees, as far as the occurrence of the top event does not depend on timing or sequencing of events.

To account for timing or sequencing of events other analysis techniques are appropriate (like state-transition diagrams, Petri Nets) which, however, are beyond the scope of these guidelines.

3

3 - FAULT TREE DESCRIPTION

3.1. Formalism / Identification / Labelling

The graphical representation of the fault tree requires that symbols, identifier and labels be used in a consistent manner.

Each event in the fault tree shall be uniquely identified. Events should be labelled so that cross reference from the fault tree to the corresponding design documentation can easily be made.

The top event of the fault tree is the undesirable event which is the primary reason for undertaking the fault tree analysis. It should be noted that only a single top event may be associated with a given fault tree (meanwhile, note that particular sub-trees of a given tree might be used to analyse specific feared events representing causes of the main top event).

If several events in a fault tree all refer to different failure modes of the same item, then such events shall be labelled so as to enable them to be distinguished. At the same time, it should be clear that they are a group of events related to the same item.

If a particular event occurs in several places in a tree, or if it occurs in several trees, all such occurrences shall bear the same label. However, events which are similar but which involve different items shall not be labelled identically.

A typical event code should contain information relating to system identification, component identification and failure mode.

The fault tree is in effect a diagram in which the events are linked by logic gates. Each gate has one output event but one or more input events. The gates show the relationship of events (causes) explaining the occurrence of a "higher" event.

The input events identify possible causes and conditions for the occurrence of the output events. However, such linking does not necessarily define the sequential (time) relationship between the events.

The basic fault tree uses AND and OR gates. However, for complex systems analysis, additional gate symbols may be required to assure that the fault trees are readable and as simple as possible.

It is important for the analyst to define and report those symbols being used and ensure uniform and consistent use throughout a given fault tree analysis. This is particularly true if computer-aided techniques are to be used.

The two basic types of fault tree gates (OR gate, AND gate), the elementary event symbol and forward events which are the most used are drawn and their meaning is given in the table below. Moreover, a complete presentation of Fault Trees standardized basic symbols is provided at APPENDIX A.

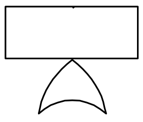
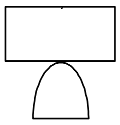
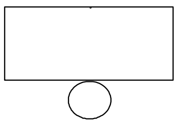
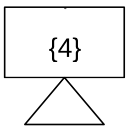
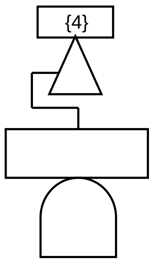
	**OR gate**	Output (event) occurs only if at least one of the input events occurs. A description of the event is provided in the rectangle.
	**AND Gate**	Output (event) occurs only if all input events occur simultaneously. A description of the event is provided in the rectangle.
	**Elementary event**	An event is defined as an elementary occurrence if it does not need further development . A description of the elementary event is provided in the rectangle.
	**Forward**	Event referring to a sub-tree – where the decomposition of this event is presented. The called decomposition (see next line) will have the same reference inside the triangle. A description of the event to which it refers is provided in the rectangle.
	**Forward target:**	Target event that will be used in another tree as a called sub-tree. The calling event (see previous line) has the same reference inside the triangle. A description of this called event (identical to that of the calling event) is provided in the rectangle.

Table 3-1: Symbols used in Fault Trees

<u>NOTE</u>: These symbols may be slightly different in other Fault Tree Analysis technique reference documents.

3.2. Boolean Logic

A fault tree can be thought as a pictorial representation of Boolean relationship among the events that cause the top event to occur. In fact a fault tree can always be translated into an entirely equivalent set of Boolean equations.

Once the fault tree has been drawn, it can be evaluated to yield its qualitative and quantitative characteristics. This is obtained from the equivalent Boolean equations.

To translate a fault tree into an equivalent set of Boolean equations, rules of Boolean algebra apply. These are listed and explained in the APPENDIX C.

The **OR-gate** represents the logical "union" of the input events. The OR gate is equivalent to the Boolean symbol " + ".

For example, the OR gate with two input events, as shown in Figure 3.1, is equivalent to the following Boolean expression:

FE = A + B

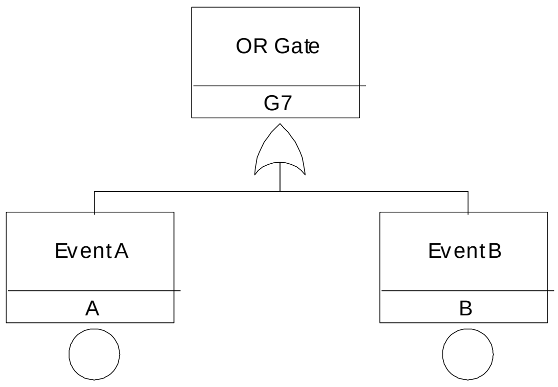


Figure 3.1: Example of an OR gate

In terms of probability:

$P(FE) = P(A) + P(B) - P(A \cap B)$

Several observations can be made about this expression:

- If A and B are mutually exclusive events, then $P(A \cap B) = 0$ and: $P(FE) = P(A) + P(B)$
- If A and B are independent events, then $P(A \cap B) = P(A)\,P(B)$ and: $P(FE) = P(A) + P(B) - P(A)\,P(B)$
- If A and B are independent, low probability events ($< 10^{-1}$), then $P(A \cap B)$ is small compared with $P(A) + P(B)$ so that $P(A) + P(B)$ is a very accurate approximation of P(FE).

NOTE: For n input events attached to the OR-gate, the equivalent Boolean expression is FE = A1 + A2 + A3 + … + An.

Consequently, if the low probability events assumption applies: P(FE) = P(A1) + P(A2) + … + P(An).

The **AND gate** represents the logical "intersection" of the input events. The AND gate is equivalent to the Boolean symbol "·".

For example, the AND gate with two input events, as shown in Figure 3.2, is equivalent to the following Boolean expression:

FE = A · B

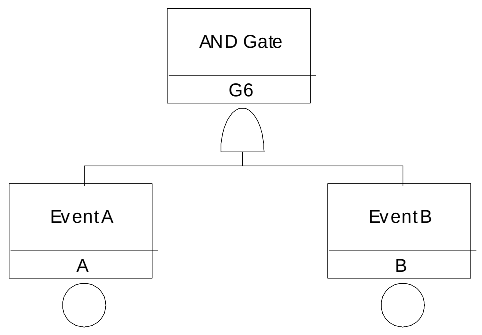


Figure 3.2: Example of an AND gate

In terms of probability:

P(FE) = P(A) P(B|A) = P(B) P(A|B)

An observation can be made about this expression:

- If A and B are independent events, then P(B|A) = P(B), P(A|B) = P(A), and P(FE) = P(A) P(B)

NOTE: For n input events attached to the AND-gate, the equivalent Boolean expression is FE = A1 · A2 · A3 · … · An.

Consequently, if the events are independent: P(FE) = P(A1) P(A2) … P(An).

IMPORTANT

Note that a necessary condition for the Fault Trees to be correctly evaluated qualitatively and quantitatively (based on the Boolean reduction) is the independency of the basic events.

Before proceeding to the qualitative & quantitative evaluation, the Fault Tree should be developed at a low enough level to ensure that condition is reached.

Moreover, be aware that Fault Trees don't allow to account for stochastic dependency (a classical example of stochastic dependency is the one of two pumps operating in active redundancy, each one charged at 50% and displaying a failure rate λ; in case of one pump failure, the remaining one will increase its charge at 100%, resulting in an increase in its failure rate $\lambda' > \lambda$). Stochastic dependency can be modelled and evaluated by using state-transition diagrams (e.g. Markov chains).

3.3. Rules for construction

Fault tree analysis proceeds in steps. Although the construction approach might have some specific aspects being function of the particular system analyzed, the approach presented in this section exhibits those fundamental steps that shall be common to all fault tree analyses.

The construction of the fault tree should be preceded by the following steps:

- Scope of analysis,
- System familiarization,
- Top event identification.

3.3.1. Scope of analysis

The definition of the scope encompasses the definition of the system boundaries, the purpose and extent of the analysis and the basic assumptions made. These assumptions should include those related to the expected operating and maintenance conditions as well as to system performance under all possible conditions of use (normal mode, degraded modes).

3.3.2. System familiarization

In order for a fault tree analysis to be carried out successfully, a detailed knowledge of the system and of its operation are required. However, some systems may be too complex to be understood fully by the safety analyst. In this case, the process of familiarization requires that the necessary specialized knowledge be obtained using expertise from competent actors and incorporated as appropriate into the fault tree analysis. The information necessary could be:

- a summary of the design intent,
- the boundaries of the system and its interfaces,
- the functional description,

- the physical structure (architecture),
- an overview of the main operating&maintenance procedures,
- the system's environmental conditions,
- a list of applicable documents (drawings, specifications, procedures, …),
- etc.

3.3.3. Top event identification and associated safety measure

The top event is the focus of the entire analysis. Such an event may be the onset or existence of a dangerous condition, or the inability of the system to provide a desired performance.

In a global safety assessment process, the top event is usually output from a previous analysis (e.g. a hazard identified by a Functional Hazard Assessment, a sub-system failure mode having resulted from a system-level Failure Modes and Effects Analysis, an event associated to a Safety Objective or Requirement that have to be met by the system, etc.). The top event is defined with respect to the mission(s) that the analyzed system must fulfil.

Generally, one or several measures are defined for a top event. The top event measure is necessary for the Fault Tree quantitative evaluation.

The **measures** represent **safety** attributes like the probability to reach an unsafe state (no dimension) or the frequency of occurrence of unsafe events (expressed, for ATM systems, in events/flight*hour or events/ATSU operational hour), or other attributes that address safety indirectly. The latter are the reliability, availability, continuity of service, integrity, maintainability or security (confidentiality).

3.3.4. Fault tree construction

Fault trees may be drawn either vertically or horizontally. If the vertical arrangement is used, the top event should be at the top of the page and the basic events at the bottom. If the horizontal arrangement is used the top event may be on the left or right of the page.

NOTE: This guidance material gives preference to the vertical arrangement.

For the construction, a systematic approach is required. To implement this systematic approach, two concepts have to be understood and used consistently. These are the concepts of "immediate cause" and of "basic unit".

The **"immediate cause"** concept requires that the analyst determines the immediate necessary and sufficient causes for the occurrence of the analyzed event (which could be the top event or an intermediary event in the tree). This concept is explained in APPENDIX B.

The immediate, necessary and sufficient causes of the top event are addressed as sub-top events and the analyst proceeds to determine their immediate, necessary and sufficient causes.

In this way, the analyst proceeds down the tree continually approaching a finer resolution.

The concept of "**basic units**" can be used to save the analyst the effort of developing fault tree diagrams which do not yield new or useful information. This concept is explained in APPENDIX B.

In the two following figures (Figure 3.3 and **Error! Reference source not found.**), two examples are presented to show the development and representation of a fault tree. Symbols used in these examples are the ones previously described (AND gate, OR gate, elementary cause symbol).

In the Figure 3.3, event A will occur only if both events B and C occur. Event C is present if either event D or E occurs.

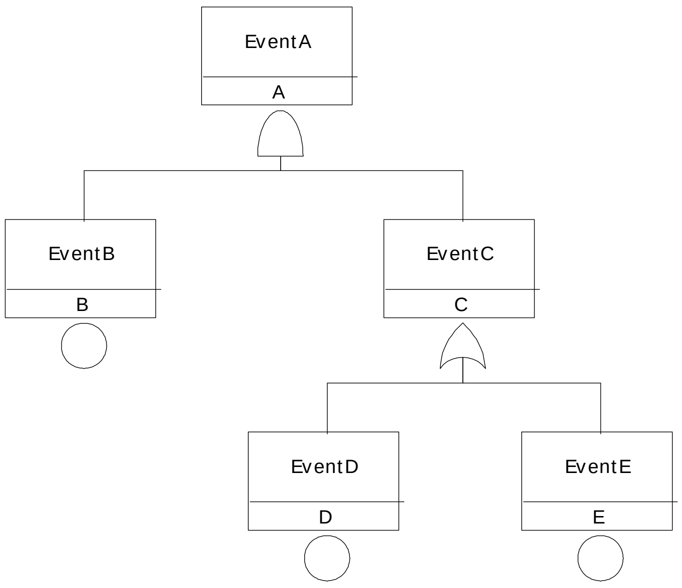


Figure 3.3: Example of a fault tree

NOTE: In this example, for each event, A, B, etc., information included in the event description box is:

- event code;
- name or description of event.

In a fault tree, common events and common cause events could appear.

<u>Common event</u> is an event which appears in different branches of a fault tree. It must display the same label and the same name or description of event.

Common cause events are elementary (or basic) events that involve a common cause of failure not made evident at the current level of detail of the analysis (e.g. several identical HW components exposed to the same temperature or humidity conditions, two replicas of the same SW application, etc.).

NOTE: A sub fault tree which appears in different branches of a fault tree could be considered as a common intermediate event.

The impact of common events is illustrated by the two following fault tree structures (see event A in Figure 3.4 and Figure 3.5) which may appear to be different; however, according to Boolean logic defined above, they are equivalent.

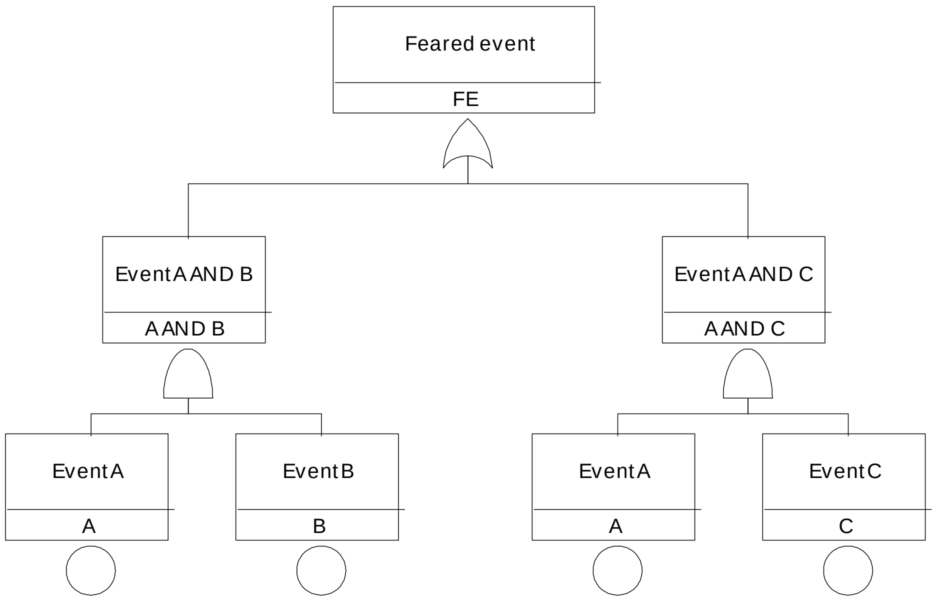


Figure 3.4: Fault tree structure 1

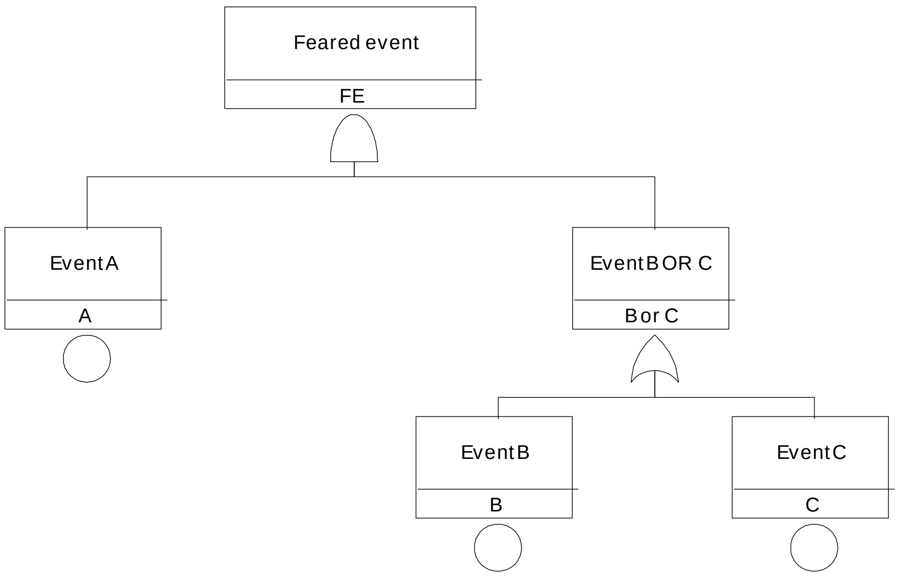


Figure 3.5: Fault tree structure 2

NOTE: There is not one "correct" fault tree for a problem but many correct forms which are equivalent to one another. The rules of Boolean algebra can thus be applied to restructure the tree to a simpler, equivalent form for ease of understanding or for simplifying the evaluation of the tree. Boolean algebra rules shall be applied to obtain the reduced form of the fault tree, called the minimal cut set form, which allows quantitative and qualitative evaluations to be performed in a straightforward manner.

3.4. Qualitative approach / Quantitative approach

The primary purposes of qualitative (logical) and quantitative (numerical) safety evaluation of a system are:

- identification of elementary events and minimal combinations thereof which can cause a system failure, and evaluation of the safety measures associated to that system failure;
- assessment of the system fault tolerance (ability to function even after a specified number of lower level failures or events contributing to the occurrence of a system failure have happened);
- verification of the independence of failure of systems, subsystems or components;
- assessment of data to locate critical components and failure mechanisms;
- identification of device failure diagnostics, inputs to repair and maintenance.

The assessment of the system fault tolerance includes a determination of the degree of redundancy in the system and verification that the redundancy is not impaired through common events or common cause events.

3.4.1. Qualitative (logical) evaluation

Two basic techniques are used for qualitative (logical) evaluation:

- Investigation;
- Boolean reduction (allowing determination of minimal cut sets);

The Boolean reduction allows further performing:

- Qualitative importance analysis;
- Common cause analysis.

Given the importance of the Boolean reduction, a brief definition and description of its use is provided in the current chapter, whilst more detailed information is available in APPENDIX C.

Investigation

Investigation includes a review of the fault tree structure, identification of common events and a search for independent branches. Investigation provides the analyst with important information which, in some cases, may be sufficient to eliminate the need for further analyses.

In all other cases, investigation is necessary for a correct decision on the type and extent of further analyses. Direct usual investigation is possible only for small trees with reduced complexity. Investigation of larger or more complex trees, as arise from the analysis of actual systems, requires a suitable software tool, but the overall approach remains the same.

Investigation deals with the review of the fault tree structure. All events which are linked to the top event through a continuous chain of OR gates are single causes for the top event to occur. Therefore, if a fault tree consists only of OR gates, no further analysis is required. If the fault tree includes other gate types, the analyzed system incorporates some sort of redundancy or other fault tolerance features which could be invalidated by common or common cause events. Investigation might identify those events, but that process becomes error prone as size or complexity of the tree increase.

An exhaustive identification of those events can be performed only after a thorough analysis using Boolean reduction (determination of minimal cut sets). As the difficulty of the analysis increases rapidly with the size of the fault tree, inspection of the fault tree allows the analyst to identify which branches of the fault tree are independent and can thus be analyzed separately.

Boolean reduction (determination of minimal cut sets)

Boolean reduction can be carried out by solving Boolean equations for the fault tree. The result are the minimal cut sets.

A cut set is a group of events which, when occurring together cause the top event to happen. A minimal cut set is the smallest such group in which all events must occur for the top event to occur. If any of the events in a minimal cut set does not occur, it prevents the top event from occurring (by this combination).

There are several methods of performing Boolean reduction and determining minimal cut sets (see APPENDIX C) but the application of some of them to large or complex trees may be difficult and incomplete. The most efficient (and as such used by FTA SW tools algorithms) are: binary decision diagrams and "prune and collapse". Various computer programs are available to assist the analyst.

Use of Minimal Cut Sets

Minimal Cut Sets are the main instrument for performing the qualitative analysis of the Fault Tree, mainly the qualitative importance analysis and the common cause analysis.

Moreover, minimal cut sets are the necessary basis for the quantitative evaluation of the Fault Tree.

During the PSSA, as an alternative to a quantitative allocation (when probabilistic data is missing) the minimal cut sets drive the process of apportioning the Safety Objectives associated to the top event into Safety Requirements allocated to the basic events.

Qualitative importance analysis

After obtaining the minimal cut sets, some idea of basic event importance (in terms of contribution to the top event) can be obtained by ordering the minimal cut sets according to their size. The single-event minimal cut sets (if any) are listed first, then the double-event minimal cut sets, then the triple, etc.

Because the failure probabilities associated with the minimal cut sets often decrease by orders of magnitude as the size of the cur sets increases, the ranking according to size gives a gross indication of the importance of the minimal cut set.

The single-event minimal cut sets (events which by occurring alone lead to the undesired event) provide essential information on the critical failures in the system. In a safety allocation process during the system design, these failures are susceptible to generate stringent safety requirements, in terms of limiting their frequency of occurrence or of limiting the severity of their effects (e.g. by providing detection and recovery means allowing to mitigate those effects).

In some cases, the double-event minimal cut sets can provide some useful information as well. If a same basic event occurs in a large proportion of these cut sets, it might be revealing a critical point in the system. Nevertheless, the qualitative analysis should be completed by a quantitative evaluation (if probabilistic data available) before driving final conclusions.

Common cause analysis

At a given point of the Fault Tree construction, highlight is needed on the potential for common causes. That information allows either to focus further efforts in developing the tree at a lower level of detail in order to explicitly reveal those common causes (as common events), or to identify potential common mode failures and make the appropriate assumptions accordingly.

The susceptibility that component failures may have a common initiating cause can be indicated by analysing the minimal cut sets. By definition, the top event occurs if all the basic events in a minimal cut set occur. So, what is interesting are only the common causes which can trigger all the basic events in a minimal cut set. A cause which does not trigger all the basic events in a minimal cut set will not by itself cause system failure.

The common cause analysis requires the identification of minimal cut sets which are susceptible to common cause failures. For doing that, common cause categories, which are general areas that cause component dependence, could be defined first. Examples of common cause categories include manufacturer, location, seismic susceptibility, flood susceptibility, wear-out susceptibility, other environment factors (temperature, humidity, radiation), operator interactions, test degradation or maintenance degradation. For each common cause category, several elements are defined (e.g. for the "location" category, the site might be divided into a number of physical locations) then when basic events are labelled, a part of the label could denote the category and element for that event. The aim is to be able to easily find the minimal cut sets whose basic events all have the same element of a given category. Finally, the most difficult task is to screen these cut sets to determine those which may require further action.

3.4.2. Quantitative (numerical) evaluation

The purpose of quantitative (numerical) evaluation of the Fault Tree is to provide a quantitative assessment of the safety measure (e.g. probability of occurrence) of the top event or of a selected set of events. Quantitative evaluation is also used to supplement the qualitative evaluation in the processes of architecture optimization, risk mitigation and Safety Requirements allocation based on Fault Trees. It uses in input the results (minimal cut sets) of the qualitative evaluation. In order to perform a numerical evaluation of a fault tree, probabilistic data at the elementary (basic) event level are required. Reliability prediction techniques, actual test or field experience data may be used to establish the input quantitative values.

If a quantitative evaluation is planned, it will be necessary to define the safety measures to be computed for the top event and to select the probabilistic data to be used in input (for the elementary events) accordingly .

In addition to the evaluation of the safety measure(s) associated to the top event, the quantitative analysis encompasses the importance analysis, the sensitivity analysis and the uncertainty analysis.

An importance analysis is used for evaluating the role of an elementary event in the occurrence of the top event.

Importance factors seek to answer the question:

- "What are the most important contributors to the safety measure associated to the top event?"

in order to identify the most efficient strategy of improving the system safety by improving the probabilistic characteristics of its elements.

Several importance factors exist (conditional probabilities, marginal importance factor; critical importance factor, diagnostic importance factor, risk increase or decrease factors, etc.)

The choice of the importance factors to be computed and their interpretation should be done with caution.

A sensitivity analysis is used for observing the variation on the probability of the top event, induced by the variation of probabilistic parameters assigned to the basic events. The variation of probabilistic parameters describes aspects like changing maintenance and checking times, implementing design modifications or changing component reliabilities. The result is a fine evaluation of the contribution of those parameters to the safety measure evaluated for the top event. Provided that adequate SW tools are available for supporting it, the sensitivity analysis should be preferred to the analysis based on importance factors.

An uncertainty analysis aims at estimating the degree of uncertainty of the safety measure associated to the top event, based on assumptions on the uncertainty associated to the probabilistic data assigned to the basic events.

More information about the importance, sensitivity and uncertainty analysis are provided in APPENDIX D.

3.4.3. Safety measures quantification

The following sub-sections are dedicated to the quantitative evaluation of the safety measure(s) associated to a Fault Tree top event.

The safety measures quantification is most easily performed in a sequential manner, first determining the basic event probabilistic characteristics (unavailability q(t), occurrence rate w(t), unreliability f(t), other probabilities, e.g. fail on demand, etc), then the measures associated to the minimal cut sets and finally the ones for the top event.

The most current safety –related characteristics to be quantified for a minimal cut set or for a top event are:

- Unavailability Q(t)
    - For minimal cut set: the probability that all the components(i.e. basic events) in the minimal cut set are down at time t and unable to operate,
    - For top event: the probability that the system is in the degraded state involved by the top event at time t and unable to operate if called on;
- Occurrence rate W(t)
    - For minimal cut set: the probability per unit time of the minimal cut set occurring,

    - For top event: probability per unit time of top event occurrence at time t

- Unreliability F(t)
    - For minimal cut set: the probability of the minimal cut set occurring on (0, t]
    - For top event: the top event occurring on (0, t]

The computation of the minimal cut set safety related characteristics is done based on the basic event probabilistic characteristics which in their turn are computed from the parameters associated to each basic event. There are several types of basic events, each of them being characterized by a specific set of probabilistic parameters. A non-exhaustive list of these types together with the relevant parameters and an approximation for calculating their probabilistic characteristics (supposing that occurrence rate is constant in time) is provided at Table 3-2.

The same table provides approximate formulas for the quantification of safety related measures for a minimal cut set and for the top event.

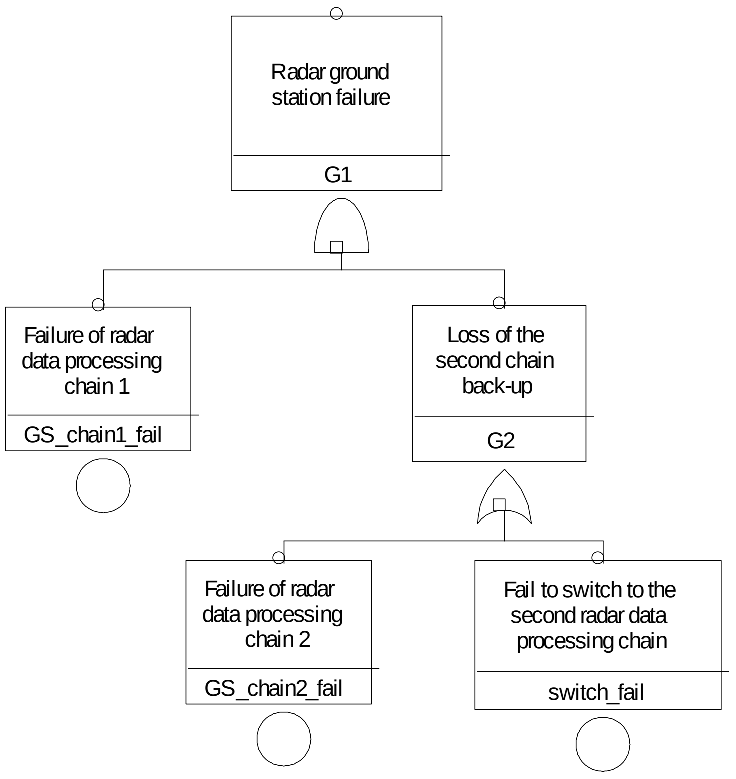
	Required data	Unavailability		Occurrence Rate	Unreliability
<u>Components</u>					
$\lambda -$*types:*					
Non-repairable	λ	$q(t) = 1 - e^{-\lambda t} \cong \lambda t, \quad where \quad \lambda t < 0.1$		$w(t) = \lambda e^{-\lambda t} \cong \lambda, \lambda t < 0.1$	$F(t) = 1 - e^{-\lambda t} \cong \lambda t, \lambda t < 0.1$
Repairable, monitored	λ , T_D	$q(t) = \dfrac{\lambda T_D}{1 + \lambda T_D} \cong \lambda T_D, \quad where \quad \lambda T_D < 0.1$		$w(t) = \lambda (asymptotic)$	same as above
Repairable, periodically tested	λ , T , T_R	$q(t) = \dfrac{\lambda T}{2} + \lambda T_R \cong \dfrac{\lambda T}{2}, \quad where \quad T_R < 0.1T$		$w(t) = \lambda (asymptotic)$	same as above
p-type					
On demand component	p	$q(t) = p$		$w(t) = 0$	F(t) = same as q(t)
<u>Minimal Cut Sets</u>		$Q_i(t) = \prod_{k=1}^{n_i} q_k(t)$	where n_i = the number of components in the i^{th} minimal cut set	$w_i(t) = q_2(t)q_3(t)...q_{n_i}(t)w_1(t)$ $+ q_1(t)q_3(t)...q_{n_i}(t)w_2(t)$ $+ q_1(t)q_2(t)...q_{n_i}(t)w_3(t)$ . . $+ q_1(t)q_2(t)...q_{n_i -1}(t)w_{n_j}(t)$	$F(t) = w_i t$
<u>System</u>		$Q_s(t) = \sum_{i=1}^{N} Q_i(t)$	where N = the number of minimal cut sets	$W_s(t) = \sum_{i=1}^{N} W_i(t)$	$F(t) = W_S t$

Where

λ	=	component failure rate per hour (operating or standby, as applicable)
T_D	=	average downtime per failure in hours
T	=	test interval in hours
T_R	=	average repair time per failure in hours
p	=	probability of cyclic component failure per demand
n(t)	=	expected number of demands in time t (note that for one demand n=1 and q(t)=p)
k(t)	=	cyclic component demand rate per hour at time t

Table 3-2: Summary of equations for approximate quantification of safety related characteristics

The practical example below illustrates the quantification of each of the safety measures above.


Radar ground station failure
G1
Failure of radar data processing chain 1
GS_chain1_fail
Loss of the second chain back-up
G2
Failure of radar data processing chain 2
GS_chain2_fail
Fail to switch to the second radar data processing chain
switch_fail

The Boolean reduction results in the following minimal cut sets:

Top= GS_chain1_fail . GS_chain2_fail + GS_chain1_fail . switch_fail

The probabilistic parameters associated to the basic events are as follows:

Component	Failure rate [events/hour]	Mean downtime [hours]	Fail on demand probability
GS_chain1_fail	1e-4 (λ)	10 (T_D)	
GS_chain2_fail	1e-4 (λ)	10 (T_D)	
Switch_fail			1e-03 (p)

The safety measures quantification will be:

Q_{top} =q(GS_chain1_fail).q(GS_chain2_fail) +q(GS_chain1_fail).q(switch_fail) =
=$\lambda.T_D.\lambda.T_D + \lambda.T_D.p$ = =2e-06

W_{top} = q(GS_chain2_fail).w(GS_chain1_fail) +
+q(GS_chain1_fail).w(GS_chain2_fail) + q(switch_fail).
w(GS_chain1_fail) =$\lambda.T_D.\lambda + \lambda.T_D.\lambda + p.\lambda$ = 3e-7

F_{top} (1000 hours)= W_{top}.t= 3e-7.1000= 3e-4

Note that the fact of confusing the quantification of the top event occurrence rate with the unavailability, involves for the example an error of one order of magnitude !

3.4.4. Rules for quantification

The quantitative evaluation can be performed for the top event, for an intermediary event or for a minimal cut set.

The following rules are to be followed to ensure that the calculations are correct:

- The safety measure to be computed should be clearly defined. It could be the probability to reach an unsafe state (no dimension) or the frequency of occurrence of unsafe events (expressed, for ATM systems, in events/flight*hour or events/ATSU operational hour), or other attributes that address safety indirectly: reliability, availability, continuity of service, integrity, maintainability, etc;

- Each data associated to a basic event must be clearly defined in terms of measure (failure rate, unavailability, probability, repair rate, mean time between failures, mean down time, etc);

- A way for ensuring that the Fault Tree is correctly evaluated is to check for the significance of safety measures associated to the different intermediary events. As an example, if the measure to be computed at the top is the frequency of occurrence of an undetected error, the intermediary event "main process is erroneous" should be measured by a frequency of occurrence, whilst a second intermediary event "detection of a main process error by the control system" combined to the former via an AND gate, should be measured by a probability (of being unavailable or of failing on demand);

- Another way to check the Fault Tree evaluation correctness is to ensure that an evaluation done for any minimal cut set, produces the same type of safety measure as the one for the top event;

3.5. Advantages of FTA

The advantages of the Fault Tree Analysis are given hereafter.

- Fault trees are particularly suited to the analysis of complex systems as nuclear power generating stations, aircraft, communication systems, chemical and other industrial processes.

- Fault Trees allow modelling of both equipment failures (HW and SW) and human errors (nevertheless, the quantitative evaluation is restricted by the usual limits imposed by the lack of probabilistic data for SW failures and human errors);

- Fault Trees are generally easier to read and to understand than other dependability models (e.g. state-transitions graphs or Petri Nets);
- Fault Trees might be less easy to build and understand than other types of models (e.g. the Reliability Block Diagrams), but far more powerful in terms of representation of failure behaviour of complex systems;
- FTA can handle multiple failures or combinations of failures;
- It can expose the needs for control or protective actions to mitigate the risk;
- It allows highlighting of the design weak points;
- The technique is well accepted and lends itself for quantification (with the limits exposed above);
- The results can provide either qualitative or quantitative data for the risk assessment process;
- Mature and powerful software tools exist to support fault trees edition and evaluation;
- FTA is a good dialogue tool for multidisciplinary teams.

3.6. Limitations/weaknesses of FTA

The limitations/weaknesses of the Fault Tree Analysis are the following ones.

- Potential exist for failure paths to be missed: dependent on contributing factors and causes adequate identification, and on sufficient exploration depth; (note that state-transition graphs ensure a more exhaustive sweeping of all possible paths of failure and recovery, but the effort of building and validating that type of models is unaffordable for complex systems);
- A fault tree may get very large and complex but some market available computer tools support this complexity; the difficulty of validating the fault tree models still remains;
- Dependent failures that occur by fault propagation (domino effects) as well as stochastic dependencies cannot be handled (unlike the states-transitions graphs);
- The modelling of human errors doesn't account for the human behaviour (need for coupling with human factors models that should generate the human errors to be taken on board as basic events in the Fault Trees);

- Dynamic (behavioural) aspects and temporal aspects, which are often encountered with SW applications, are not covered by Fault Trees. Other methods allowing to cover those aspects are recommended to be used in complement to Fault Trees (e.g. the use of Event Trees allows to account for sequencing of events and can be coupled easily to Fault Trees; the combination of these techniques is the basis of the nuclear safety assessment models- PSA) . To model more complex behaviour (e.g. synchronisation, concurrency, parallelism, etc), other techniques should be employed (like states-transition graphs and Stochastic Petri Nets);

- The method concentrates its attention to specific top events; if these events were not specified adequately or not in an exhaustive manner, other serious consequences of malfunction might not be revealed. Other methods, like the states-transition graphs or Stochastic Petri Nets, don't have this drawback, but in exchange they generate huge models, difficult to build and validate.

4

4 - FTA IN SUPPORT OF THE PSSA

4.1. PSSA Safety Requirement Specification process

As specified in the SAM Methodology, the PSSA SRS is a five-stage process, as shown in Figure 4.1:

Stage 1. Refine sub-functions safety contribution,

Stage 2. Evaluate system architectures,

Stage 3. Apply risk mitigation strategy,

Stage 4. Apportion safety objectives into safety requirements to system elements,

Stage 5. Balance / reconcile safety requirements.

The FTA technique provides valuable support to all these stages. The fault trees might be progressively built and refined along these stages.

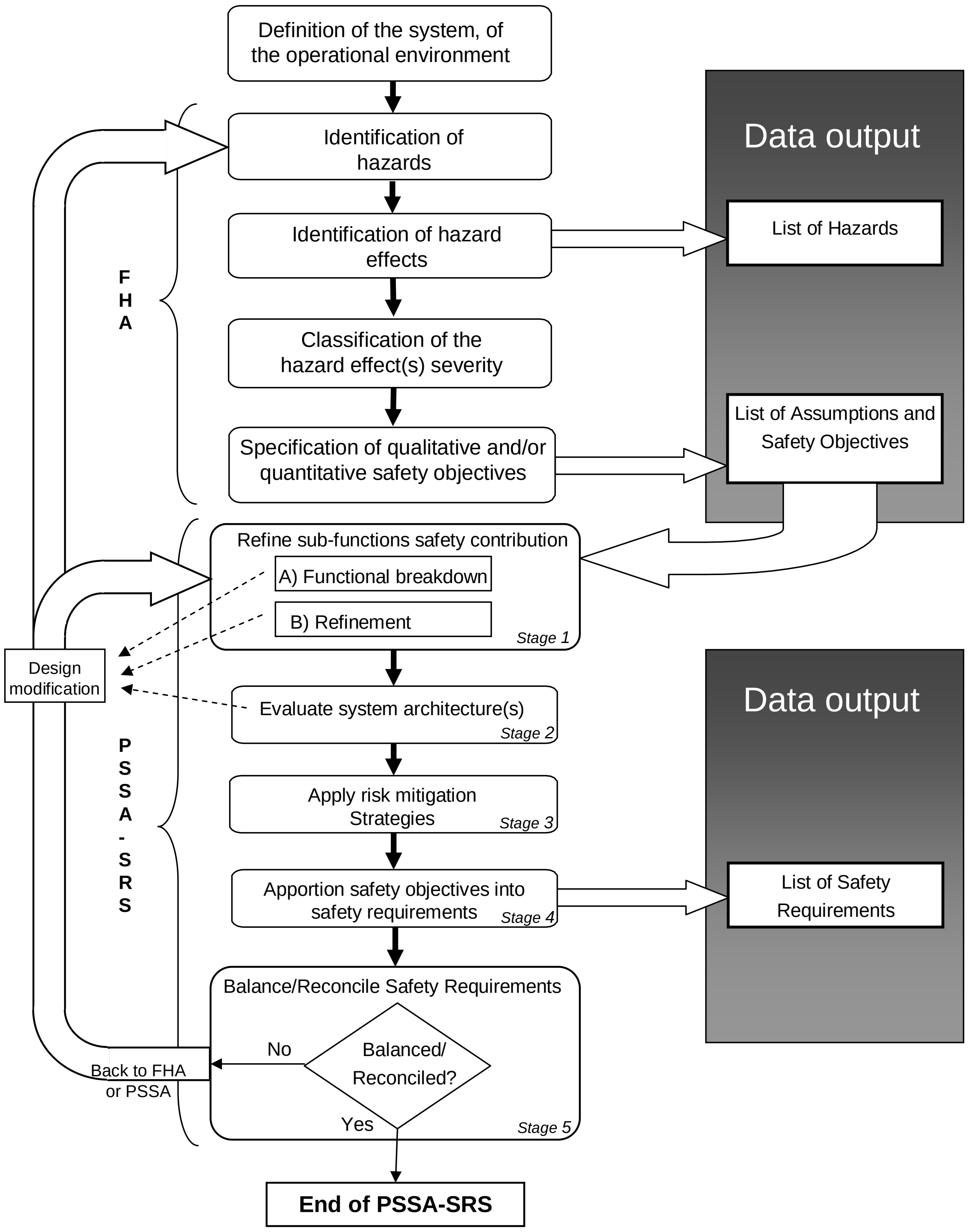

Definition of the system, of the operational environment
Identification of hazards
Identification of hazard effects
Classification of the hazard effect(s) severity
Specification of qualitative and/or quantitative safety objectives
FHA
Data output
List of Hazards
List of Assumptions and Safety Objectives
Refine sub-functions safety contribution
A) Functional breakdown
B) Refinement
Stage 1
Design modification
Evaluate system architecture(s)
Stage 2
Apply risk mitigation Strategies
Stage 3
Apportion safety objectives into safety requirements
Stage 4
PSSA-SRS
Data output
List of Safety Requirements
Balance/Reconcile Safety Requirements
No
Balanced/ Reconciled?
Yes
Stage 5
Back to FHA or PSSA
End of PSSA-SRS

Figure 4.1: Safety Requirements Specification Process

4.2. Required Inputs

The essential pre-requisite and inputs for conducting a PSSA using Fault Tree Analysis are:

- a system definition:
    - refinement of the functional analysis performed in the FHA, by identifying lower level sub-functions, their relationships and interaction with the environment;
    - assumptions;
    - list of hazards and their associated safety objectives, as output by the FHA;
- a system design (or design intent):
    - description of system architecture(s) and their rationale;
    - interfaces and boundaries;
    - design constraints;
    - system elements requirements and/or specification;
- a description of the system operations and environment
- regulatory requirements;
- applicable standards.

All of these inputs or most of them, depending on the analysis to be done, are necessary to develop the level of understanding of the system design and its rationale required by the FTA performance.

4.3. FTA for refining sub-functions safety contribution

This task is related to the life cycle stage of definition (or refinement) of the system functional architecture: high level functions identified during the System Definition phase are successively decomposed into lower-level sub-functions. The functional breakdown is pursued until each sub-function becomes sufficiently defined to be allocated to a system element: human, procedure or equipment (hardware, software) and such ensuring passage to the next life cycle stage: architecture design.

The safety assessment to be carried on along the stage of definition of system functional architecture is aimed at refining the sub-functions contribution to safety, starting from the global safety contribution identified by the previous safety assessment step (FHA).

For doing that, a Fault Tree Analysis might be performed until the sub-functions decomposition level is reached (note that generally the FHA addressed the failure of services or user-oriented functions; the role of the current FTA is to analyze the failure of lower level technical functions that implement the former).

The objective for the safety assessment is to filter-out the system (sub)functions with no safety implication in order to retain for the next steps (involving an in-depth safety assessment) only the (sub)functions displaying a safety contribution.

The FTA is to be complemented by a functional Failure Modes and Effects Analysis (FMEA). These two techniques permit to accede to different results in order to refine functions. They can be used in two different ways:

Functional FMEA then FTA:

In this case, the FMEA has a great interest as a preliminary study to FTA, facilitating the construction of this latter.

It is performed at the level of decomposition of the sub-functions.

Note that generally the FHA addressed the failure of services or user-oriented functions; the role of the current FMEA is to analyze the failure of lower level technical functions that implement the former.

The functional FMEA might be complemented by functional Fault Trees, as the former technique doesn't allow to account for combination of failures, whilst the latter does.

FTA then Functional FMEA:

In this case, FTA is first used to refine functions (in sub-functions).

Functional FMEA is better indicated to refine sub-functions which are made with no redundant constituents.

For example, a functional FMEA, after an FTA is highly recommended for software functions study.

The FTA for refining sub-functions safety contribution is better indicated for:

- redundant sub-functions,
- many interactions between sub-functions.

4.4. FTA for evaluating System Architecture(s)

This safety assessment stage consists of determining if and how the architecture(s) and its (their) elements could cause or contribute to the identified hazards. When alternative architectures are proposed, the advantages/draw-backs with respect to each variant are identified and safety-related advice to support architecture choice is provided.

The FTA is one of the techniques appropriate for this purpose.

The FTA can permit to validate some assumptions of design … and not to validate others ones.

The FTA is rather for eliminating/rejecting architectures that "obviously" do not satisfy certain requirements as for example:

- isolation criteria which can be a common mode;
- level of stringency of certain requirements (failure rate …).

The fault tree(s) to be developed during this stage is the one begun during the previous stage (See chapter 4.3). It is carried on and refined in order to evaluate the system architecture.

It is first based on the qualitative analysis, and then on a quantitative analysis with data derived from experience feedback, field experience, similar systems, expert judgment, reliability database, etc. … when existing.

The use of data in the quantitative analysis can be complemented by a sensitivity analysis.

The fault trees can be constructed as explained in the first part of this guidance material (chapter **Error! Reference source not found.**).

The more details included in the fault tree and the more the nodes are refined, the more information is available to the designer for mitigating the associated risk.

It is therefore recommended to focus on exploring hazards having the most stringent safety objective.

4.5. FTA application to Risk Mitigation Strategies

The FTA could be used as a support to:

- the assessment of the efficiency of a strategy (for example taking into account a common mode or not);
- the choice of a strategy among several.

Note that the Risk Mitigation Strategies can be determined using other documents as "Guidelines for the Safety Assessment of ATM procedures (SAAP)" or or Guidance Material A of PSSA – Chapter 3, etc.

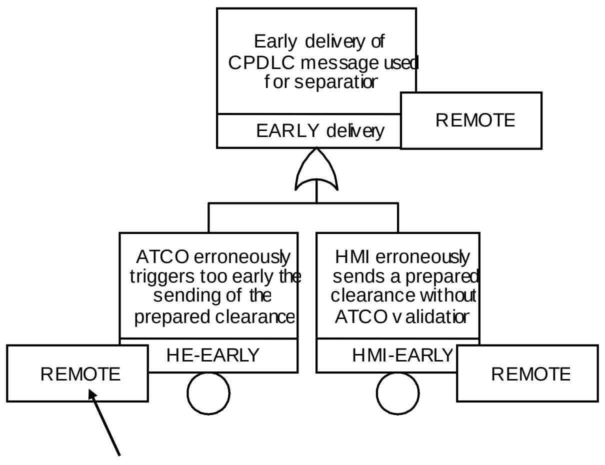
<u>EXAMPLE:</u>

In this fictitious example, before realizing the apportionment of Safety Objectives for a system, Risk Mitigation Strategies are defined as, for example, the following one:

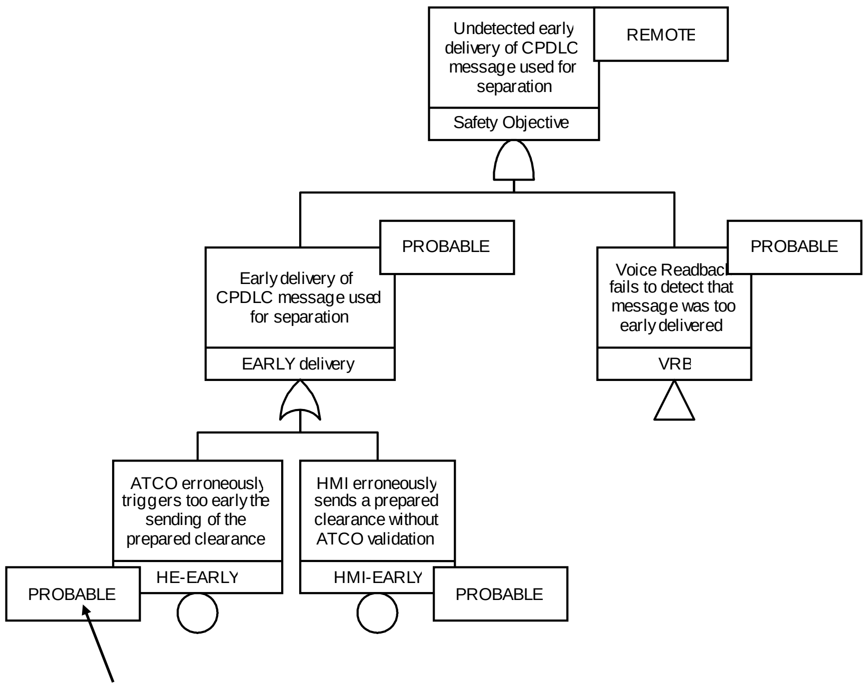
- *No human error shall be associated to a Safety Requirement whose qualitative allocation is more stringent than Probable.*

So, it means that after the first apportionment iteration, if a safety requirement allocation to a human error is more severe than Probable then the apportionment needs to be re-evaluated in order to satisfy the defined strategy.

In the following fault tree, after the first apportionment iteration, a "Remote" requirement is allocated to a human error (HE-EARLY). This does not follow the previous defined rule.


Early delivery of CPDLC message used for separation
EARLY delivery
REMOTE
ATCO erroneously triggers too early the sending of the prepared clearance
HE-EARLY
REMOTE
HMI erroneously sends a prepared clearance without ATCO validation
HMI-EARLY
REMOTE

So, one solution (described in the above figure) can be proposed in order to allocate a Probable requirement to the human error (HE-EARLY event), instead of the previous "Remote" requirement.


Undetected early delivery of CPDLC message used for separation
REMOTE
Safety Objective
PROBABLE
Early delivery of CPDLC message used for separation
EARLY delivery
Voice Readback fails to detect that message was too early delivered
PROBABLE
VRB
ATCO erroneously triggers too early the sending of the prepared clearance
HE-EARLY
PROBABLE
HMI erroneously sends a prepared clearance without ATCO validation
HMI-EARLY
PROBABLE

As it can be seen, in this case, the solution is to add detection realized with a Voice Read-Back action

Warning: this is a fictitious example. Therefore it does not suggest that Voice-read Back shall be always used for Controller Pilot Data Link Communications (CPDLC) applications.

The document "Safety and Performance Requirements Standard For Initial Air Traffic Data Link Services In Continental Airspace" (ED120) provides actual guidance for such application.

4.6. FTA for apportioning Safety Objectives into Safety Requirements

As shown in Figure 4.2, the PSSA Safety Requirement Specification (SRS) process is an iterative one.

It includes the Safety Objectives apportionment into Safety Requirements process which is a part of a local iterative one (including the Balancing / Reconciliation step).

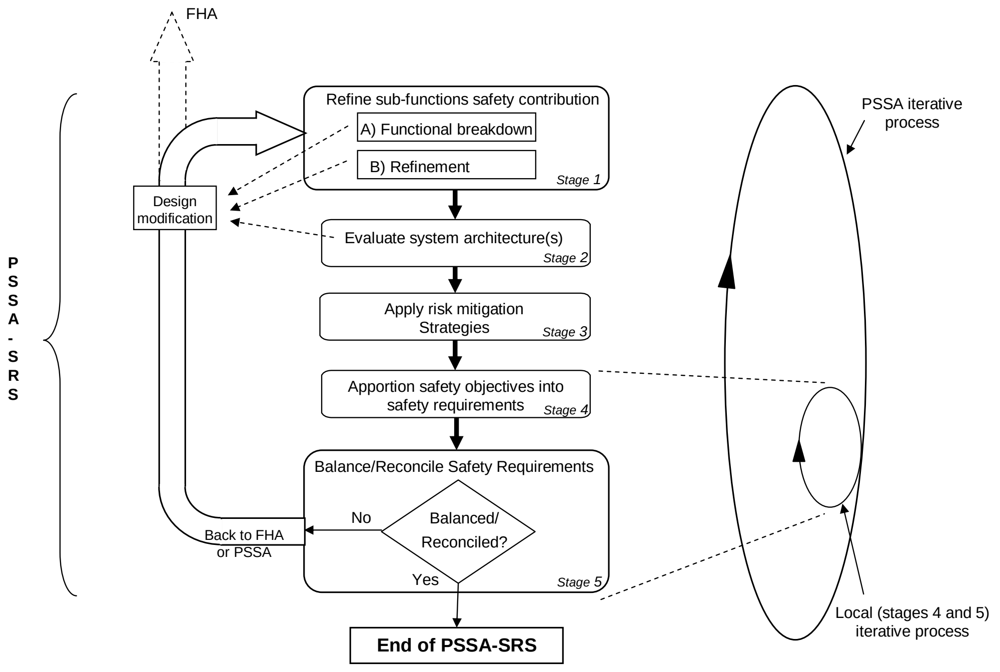

FHA
Refine sub-functions safety contribution
A) Functional breakdown
B) Refinement
Stage 1
Design modification
Evaluate system architecture(s)
Stage 2
Apply risk mitigation Strategies
Stage 3
Apportion safety objectives into safety requirements
Stage 4
Balance/Reconcile Safety Requirements
No
Balanced/ Reconciled?
Yes
Stage 5
Back to FHA or PSSA
End of PSSA-SRS
PSSA-SRS
PSSA iterative process
Local (stages 4 and 5) iterative process

Figure 4.2: PSSA and local iterative processes

After the previous three stages of the PSSA SRS process, the fourth stage performs the apportionment of Safety Objectives into Safety Requirements for each individual system element (people, procedures and equipment).

This step and the subsequent balancing / reconciliation with the non-safety related requirements are required to ensure that the system would be able to meet the safety objectives defined in the FHA. They compose a local iterative process.

The iterative allocation of Safety Requirements might require doing one or several iterations, at local level, depending on the apportionment strategy to be used (top/down, bottom/up) and on the balancing / reconciliation step.

The iterative allocation process has to be carried on until the final set of Safety Requirements (taking into account the architecture and all the inputs) can reasonably be expected to achieve the Safety Objectives.

If the Safety Objectives cannot be achieved, the iterative allocation of Safety Requirements might require reviewing the risk mitigation strategies and even the system architecture design. This is included in the PSSA SRS iterative process.

Note that before beginning the apportionment task, the fault tree, constructed in the previous stages, needs to be developed trying to give a detailed and comprehensive overview of the contribution of all domains (Ground / Airborne) and types of element (equipment, human factor, procedure). Fault tree is elaborated by searching the causes and contributions thereof leading to the top event (as seen in the chapter Fault Tree Description). If needed, architectural FMEA (as opposed to functional FMEA) might be used to contribute to the thorough fault tree construction.

NOTE: Considering an operational failure, different kinds of causes have to be considered: causes from the current operation and causes from a previous operation that are not detected or not mitigated. This can be "modelled" in a specific form of fault tree: phased fault tree (see reference document Ref. 2).

Two apportionment strategies could be recommended:

- A "top-down" strategy, which aims at apportioning the high level Safety Objective into low level Safety Requirements, accounting for the data available, which represent the constraints. This strategy is generally appropriate when the analyzed system is a new one (from scratch) and not much data related to fault tree events is available.

- A "bottom-up" strategy, which aims at filling the missing safety data in the fault tree (by "guessing" the Safety Requirements to be allocated to the elementary events) in order to reach the Safety Objective at the top of the tree. This strategy is adequate in case of changes brought to an already existing system, when most of the safety data related to fault tree events is already known. This strategy might require more iterations than the previous one.

The top-down apportionment could be done using two approaches:

- A "step-by-step" apportionment progressing through the Fault Tree starting from the Safety Objective associated to the top event, via the intermediary events, down to the elementary events. That approach might be efficient in the case of fault trees displaying few common or common cause events. A particular application are the SW fault trees aimed at allocating SW assurance levels to the SW modules, based on the apportionment of the Safety Objective into quantitative Safety Requirements for system functions implemented by SW. Moreover, it can be adapted to those situations where quantitative probabilities are replaced by qualitative levels of probability (Rare, Occasional, etc).

- A minimal cut sets driven apportionment, performed once the Boolean reduction of the tree was accomplished. That approach might be appropriate in the case of fault trees displaying strong dependencies (common or common cause events). As that approach might need to make use of importance factors calculation, it works only if quantitative values are assigned to all the elementary events of the tree (no qualitative levels of probability).

Apportionment should account for any available data related to fault tree events:

- field experience;
- similar component failures addressed by industry reliability data bases;
- human errors addressed by industry reliability data bases;
- state-of-the art for ensuring a certain level of reliability/safety for a certain type of system element;
- availability;
- integrity;
- maintainability, etc.

Note that quantitative safety objectives result, through allocation process, into Safety Requirements addressing reliability, availability, integrity, maintainability.

Quantitative Safety Requirements might be deterministic or probabilistic:

- **Deterministic**: time to switch-over, maximum tolerable time of service interruption, maximum tolerable time for a maintenance intervention, etc.;
- **Probabilistic**: safety (free from accidents), reliability (mission success or continuity of proper service), availability (readiness for use), integrity (correctness of data), maintainability (ability to be maintained).

4.7. FTA application to balance – reconcile Safety Requirements

FTA supports design decision making process.

4.8. Required Outputs

The report of the fault tree analysis should include as a minimum the basic items listed below.

Additional and supplementary information may be provided to increase clarity, especially in cases of complex systems analyses.

Basic items in the report should be:

- Objective and scope;
- System definition / description;
- System design (or design intent):
  - description of system architecture(s) and their rationale;
  - interfaces and boundaries;
  - design constraints;
  - system elements requirements and/or specification;
- Description of the system operations and environment;
- Rationales and Assumptions having led to the FTA *;
- Fault Tree structure *;
- Results of Qualitative analysis *:
  - minimal cuts;
  - qualitative importance;
  - common cause analysis;
- Results of Quantitative assessment *;
  - List of data (with their source references);
  - Quantitative assessment for the top event;

  - Quantitative assessment for intermediary events (if necessary);
  - Importance factor;
  - Sensitivity analysis;

- Conclusions *.

* Note that the marked items are to be detailed for each hazard to be studied.

5

5 - FTA IN SUPPORT OF THE SSA

5.1. Contribution to EATMP Safety Assessment Methodology - SSA

As specified in the SAM methodology, the SSA Safety Assurance & Evidence Collection step has two objectives:

- to collect evidences;
- to provide assurance that:
  - each system element as implemented meets its Safety Requirements;
  - the system as implemented satisfies its Safety Objectives;
  - the systems satisfies users expectations with respect to the Safety.

The FTA technique provides valuable support for the "providing safety assurance" step.

5.2. Required Inputs

The essential pre-requisite and inputs for conducting a SSA using Fault Tree Analysis are:

- a system definition:
    - validation of the work performed in the PSSA (functions, sub-functions, system elements, their relationships and interaction with the environment);
    - definitive/validated assumptions;
    - list of hazards and their associated safety objectives, as output by the FHA;
- a definitive system design:
    - description of system architecture(s) and their rationale;
    - interfaces and boundaries;
    - design constraints;
    - system elements requirements and/or specification;
- a description of the system operations and environment;
- validated PSSA results (fault tree structure, qualitative and quantitative analysis, etc.);
- Safety Requirements allocated by PSSA;
- regulatory requirements;
- applicable standards.

All of these inputs or most of them, depending on the analysis to be done, are necessary to complete the level of understanding of the system design and its rationale required by the FTA performance.

5.3. FTA for Safety Objective satisfaction verification

Based on the results of the PSSA, the FTA, during the SSA process, can be used for two tasks:

- to realize a more detailed analysis for some events (or each event) linked to the system elements (human, procedure, equipment);
- to verify that, knowing quantitative data or qualitative safety results and their associated evidences, the Safety Objectives are met (bottom-up method).

5.3.1. FTA to realize a more detailed analysis of events

In this case, the aim of the use of FTA is to detail the causes of an event in order to precise the research of data and to ease the quantitative evaluation.

It could be completed with the FMEA method, at the architecture/equipment level, for the definition of quantitative values: λ, μ, MTBF, MTTR, etc.

5.3.2. FTA to verify that Safety Objectives are met

The most useful FTA utilization during SSA is to verify that the Safety objectives (at hazard level) are satisfied.

After detailing the fault tree (if necessary) and collecting data, a bottom-up method is to be performed (as explain in APPENDIX E).

It consists in giving a qualitative or quantitative value for each event defined during the PSSA step, and then in associating the values through the fault tree taking into account the logical gates or the minimal cuts.

Note that all the values are to be either qualitative ones or quantitative ones, corresponding to the characteristic of the Safety Objectives to be met.

Note that the FTA is not used to research and define data and the associated evidences.

Note that if a quantitative assessment is realized to verify the Safety Objectives satisfaction, the rules defined in the chapter **Error! Reference source not found.** are to be applied.

If the Safety Objective is not satisfied, the FTA can also be suitable for the research of what is (are) the cause(s) for this dissatisfaction.

5.4. Required Outputs

The report of the fault tree analysis should include as a minimum the basic items listed below.

Additional and supplementary information may be provided to increase clarity, especially in cases of complex systems analyses.

Basic items in the report should be:

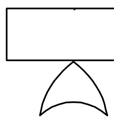
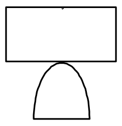
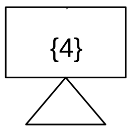
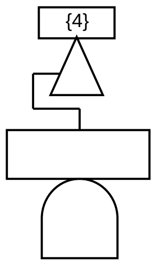
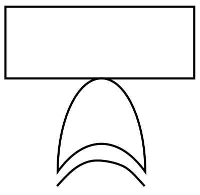
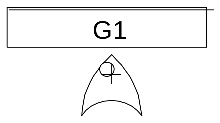
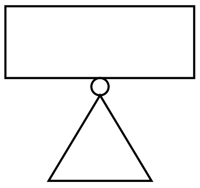
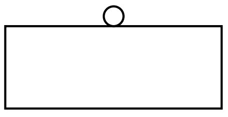
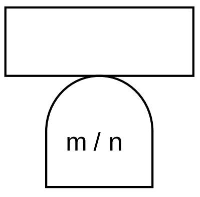
- Objective and scope;
- System definition / description **;
- System design **:
  - description of system architecture and their rationale;
  - interfaces and boundaries;
  - design constraints;
  - system elements requirements and/or specification;
- Description of the system operations and environment **;
- Rationales and Assumptions having led to the FTA **;
- Fault Tree structure *;
- Results of qualitative "Safety Objective satisfaction" analysis *;
- Results of quantitative "Safety Objective satisfaction" analysis *:
  - List of data (with their source references);
  - Quantitative assessment for the top event;
- Investigation report if the Safety Objective is not met (if necessary) *;
- Conclusions *.

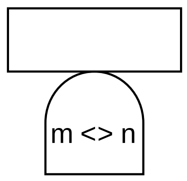
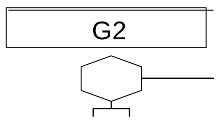
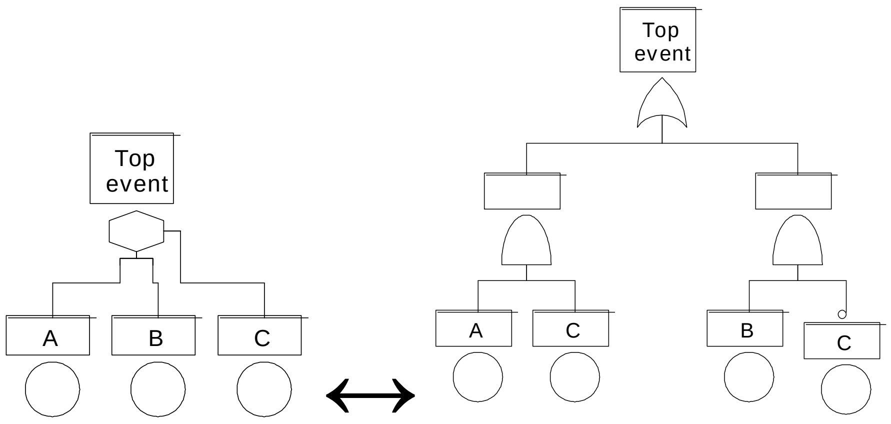
* Note that the marked items are to be detailed for each hazard to be studied.

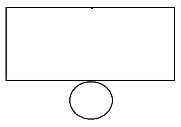
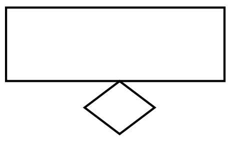
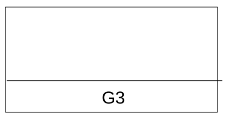
** Note that all the required outputs can be only about the discrepancies between the PSSA and SSA phases.

A

APPENDIX A STANDARDIZED BASIC SYMBOLS

	**OR gate** Output (event) occurs only at least one of the input events occurs. A description of the event is provided in the rectangle.
	**AND Gate** Output (event) occurs only if all input events occur simultaneously. A description of the event is provided in the rectangle.
	**Forward** Event referring to a sub-tree – where the decomposition of this event is presented. The called decomposition (see next line) will have the same reference inside the triangle. A description of the event to which it refers is provided in the rectangle.
	**Forward target:** Target event that will be used in another tree as a called sub-tree. The calling event (see previous line) has the same reference inside the triangle (the calling event is presented in this particular example as beginning with an "OR" gate). A description of this called event (identical to that of the calling event) is provided in the rectangle.
 or 	**Exclusive-OR gate** Output (event) occurs only if one of the input events occurs alone (used typically with two input events). A description of the event is provided in the rectangle.
 or 	**NOT gate** Output (event) represents a condition which is an inverse of the condition defined by the input event. A description of the event is provided in the rectangle. The NOT gate renders the objects selected in the tree negative (or vice versa): an AND gate becomes NAND, an OR gate becomes NOR, an event becomes a complementary event.
	**Combination gate** Output (event) occurs only if at least m of the n inputs occur simultaneously (with m < n).

	**Cardinality gate** Output (event) occurs only if x inputs occur, x comprised between M (min) and N(max).
	**IF … ELSE gate** The output (event) of the IF …ELSE gate corresponds to the event on the left (if condition verified Gate) IF the Condition event is at true ELSE it corresponds to the event on the right (if condition not verified Gate). This gate is used for representing a structure of the type If Evt1 then Evt2 else Evt3. This type of gate has been introduced to represent a twin-input switch: The gate IF …ELSE can always be represented by AND, OR, and NEGATIVE gates: 

	**Elementary event** An elementary event is defined as an elementary occurrence if it does not need further development. A description of the elementary event is provided in the rectangle.
	**Undeveloped event** An undeveloped event is an event for which further subdivision was not done (usually because it was considered unnecessary)
	**Event description block** This element is a gate allowing a comment to be inserted into a tree. It can be placed at any point whatever of the tree (for instance at the top of the tree).

B

APPENDIX B FAULT TREE CONSTRUCTION FUNDAMENTALS

B.1 The "Immediate Cause" Concept

In a Fault Tree Analysis, for an analyst, the first step is to define the system to be studied (i.e., determine its boundary) and then select a particular system failure mode for further analysis. The latter constitutes the top event of the system analyst's fault tree. ➔ This is done in the FHA.

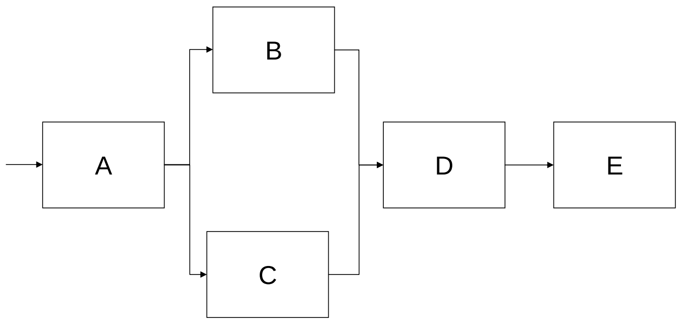
The second step is to construct the fault tree: This construction requires few fundamentals rules which are explained hereafter.

The system failure mode defined in the first step constitutes the fault tree top event. Knowing this, the analyst determines the immediate, necessary, and sufficient causes for the occurrence of the top event. It should be noted that these are not the basic causes of the event but the <u>immediate</u> causes or <u>immediate</u> mechanisms for the event. This is an extremely important point which will be clarified and illustrated in later examples.

The immediate, necessary, and sufficient causes of the top event are now treated as sub-top events and the analyst proceeds to determine their immediate, necessary, and sufficient causes.

In this way, the analyst proceeds down the tree, continually approaching further resolution, until ultimately, the limit of resolution of the tree is reached.

As an example of the application of the "immediate cause" concept, consider the simple system below.



This system is supposed to operate in the following way: a signal to A triggers an output from A which provides inputs to B and C. B and C then pass a signal to D which finally passes a signal to E. A, B, C and D are dynamic subsystems. Furthermore, subsystem D needs an input signal from either B or

C or both to trigger its output to E. There is a redundancy in this portion of the system.

This system can be interpreted quite generally. For example, it could represent an electrical system in which the subsystems are analog modules (e.g. comparators, amplifiers, etc.) ; it could be a piping system in which A, B, C and D are valves; or it could represent a portion of the "chair of command" in a corporation.

The top event to be studied is "no signal to E".

An assumption is the transmitting devices (passive components), which pass the signals from one subsystem to another, are neglected. This is tantamount to assigning a zero failure probability to the wires, pipes, or command links.

The step-by-step analysis of the top event is the following:

The immediate cause of the top event, "no signal to E," is "no output from D." The analyst should strongly resist the temptation to list the event, "no input to D" as the immediate cause of "no signal to E." In the determination of immediate causes, one step should be taken at a time. The "immediate cause" concept is sometimes called the "Think Small" Rule because of the methodical, one-step-at-a-time approach.

The sub-top event, "no output from D," is now identified and it is next necessary to determine its immediate cause or causes. There are two possibilities:

(1) "There is an input to D but no output from D."

(2) "There is no input to D."

Therefore, the sub-top event, "no output from D," can arise from the union of the two events, 1 or 2.

<u>NOTE</u>: The reader should note that if more than one step was taken and the cause of "no input to D," had been identified (improperly) then event 1 above would have been missed. In fact, the motivation for considering immediate causes is now clear: it provides assurance that no fault event in the sequence is overlooked.

The immediate causes for the new mode failures, events 1 and 2 can now be sought. If the limit of resolution is the subsystem level, then event 1 (which can be rephrased, "D fails to perform its proper function due to some fault internal to D") is not analyzed further and constitutes a basic input to the tree. With respect to event 2, its immediate, necessary and sufficient cause is "no output from B and no output from C," which appears as an intersection of two events, i.e.,

2 = 3 and 4

Where

3 = "no output from B" and

4 = "no output from C"

NOTE: As a matter of terminology, it is convenient to refer to events as "faults" if they are analyzed further (e.g., event 2). However, an event such as 1 which represents a basic tree input and is not analyzed further is referred to as a "failure."

The analysis is now carried on focusing out attention on events 3 and 4.

As far as 3 is concerned:

3 = 5 or 6

Where

5 = "input to B but no output from B" and

6 = "no input to B"

Event 5 is identified as a failure (basic tree input). Event 6 is a fault which can be analyzed further. The event 4 is studied in an analogous way.

The further steps in the analysis of this system can now be easily supplied by the reader. The analysis will be terminated when all the relevant basic tree inputs have been identified. In this connection, the event "no input to A" is also considered to be a basic tree input.

The analysis of the top event ("no input to E") consequently produced a linkage of fault events connected by "and" and "Or" logic. The framework (or system model) on which this linkage is "hung" is the fault tree.

The following section provides the necessary details for connecting the fault event linkage to its framework (fault tree),

B.2 Basic Rules for fault tree construction

The construction of fault trees is a process that has evolved gradually over a period of about 15 years. In the beginning it was thought of as an art, but is was soon realized that successful trees were all drawn in accordance with a set of basic rules.

Observance of these rules helps to ensure successful fault trees so that the process is now less of an art and more of a science.

The basic rules for successful fault tree analysis are now examined.

The fault tree in the Figure 5.1 is considered as a simple fault tree or perhaps a part of a larger fault tree.

Note that none of the failure events have been "written in"; they have been designated just Q, A, B, C, D.

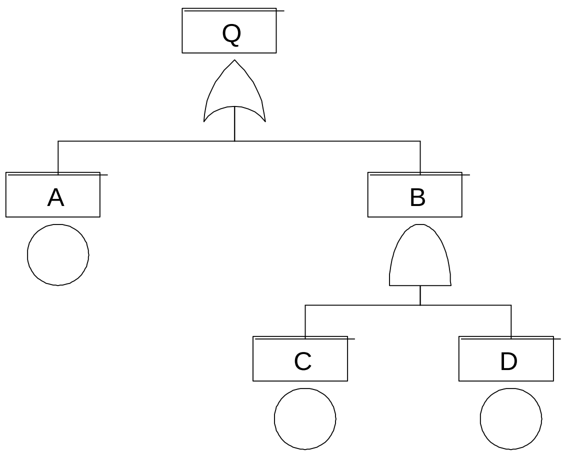


Figure 5.1: Simple fault tree

Now, when there is a specific problem to solve, it becomes necessary to describe exactly what such events such as Q, A, B, C, and D actually are, and the proper procedure for doing this constitutes **Ground Rule I**:

Write the statements that are entered in the event boxes as faults; state precisely what the fault is and when it occurs.

The "what-condition" describes the relevant railed (or operating) state of the component. The "when-condition" describes the condition of the system – with respect to the component of interest - which makes that particular state of existence of the component a fault.

Note that Ground Rule I may frequently require a fairly verbose statement. So be it. The analyst is cautioned not to be afraid of wordy statements. Do not tailor the length of your statement to the size of the box that you have drawn. If necessary, make the box bigger. It is permissible to abbreviate words but resist the temptation to abbreviate ideas. Examples of fault statements are:

(1) Normally closed relay contacts fails to open when EMF is applied to coil.

(2) Motor fails to start when power is applied.

The next step in the procedure is to examine each boxed statement and ask the question: "Can this fault consist of a component failure?" This question and its answer lead us to **Ground Rule II**:

If the answer to the question, "Can this fault consist of a component failure?" is "Yes," classify the event as a "state-of-component fault." If the answer is "No," classify the event as a "state-of system fault."

If the fault event is classified as "state-of-component," add an OR gate below the event and look for primary, secondary and command modes. If the fault event is classified as "state-of-system," look for the minimum necessary and sufficient immediate cause or causes. A "state-of-system" fault event may require an AND gate, an OR gate, an INHIBIT gate, or possibly no gate at all. As a general rule, when energy originates from a point outside the component, the event may be classified as "state-of-system."

In addition to the above ground rules, there are a number of other procedural statements that have been developed over the years. The first of these is the **No Miracles Rule**:

If the normal functioning of a component propagates a fault sequence, then it is assumed that the component functions normally.

We might find, in the course of a system analysis, that the propagation of a particular fault sequence could be blocked by the miraculous and totally unexpected failure of some component. The correct assumption to make is that the component functions normally, thus allowing the passage of the fault sequence in question.

However, if the normal functioning of a component acts to block the propagation of a fault sequence, then that normal functioning must be defeated by faults if the fault sequence is to continue up the tree. Another way of stating this is to say that, if an AND situation exists in the system, the model must take it into account.

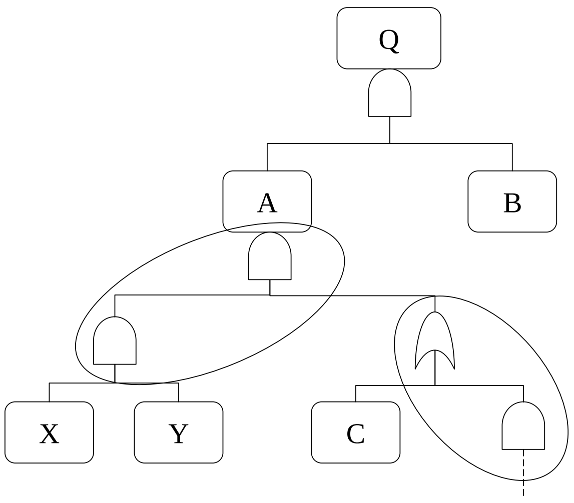
Two other procedural statements address the dangers of not being methodical and attempting to shortcut the analysis process. The first is the **Complete-the-Gate Rule**:

All inputs to a particular gate should be completely defined before further analysis of any one of them is undertaken.

The second is the **No Gate-to-Gate Rule**:

Gate inputs should be properly defined fault events, and gates should not be directly connected to other gates.

The Complete-the-Gate Rule states that the fault tree should be developed in levels, and each level should be completed before any consideration is given to a lower level. With regard to the No Gale-to-Gate Rule, a "shortcut" fault tree is shown below.



The "gate-to-gate" connection is indicative of sloppy analysis. The "gate-to-gate" shortcutting may be all right if a quantitative evaluation is being performed and the fault tree is being summarized. However, when the tree is actually being constructed, the gate-to-gate shortcuts may lead to confusion and may demonstrate that the analyst has an incomplete understanding of the system. A fault tree can be successful only if the analyst has a clear and complete understanding of the system to be modelled.

B.3 When the fault tree development should be stopped? Concept of "Basic units"

A basic unit is treated as if it were a single unit or component or dealt with separately.

In order for the unit to be considered "basic", it is necessary and sufficient that the following requirements be satisfied:

- the basic events can be sorted in function of the responsibility domain (for example Aircraft or ATC);
- the basic events can be linked to a specific type of events (for example: Hardware, Software, Human, Procedure or Relationship between two of them);
- the basic events are independent;
- both the functional and physical boundaries shall be clearly defined;
- operation of the unit shall not depend on any supporting function (shared with other units), or if it does, all events related to the unit shall be expressed by a single OR gate having one of the inputs representing a fault of the unit, the remaining inputs representing inability to perform the corresponding support functions;
- no event shall be related to a part within the unit that appears elsewhere in the fault tree.

Note that the dependency between basic events is to be validated by the Common Cause Analysis at a lower level when the PSSA is to be realized by a sub-contractor or an equipment manufacturer.

C

APPENDIX C BOOLEAN ALGEBRA / MINIMAL CUT SET / QUALITATIVE APPROACH

C.1 Rules of Boolean Algebra

The Boolean rules proposed hereafter have immediate practical importance in relation with fault trees. A fault tree can be thought of as a pictorial representation of those Boolean relationships among fault events that cause the top event to occur. In fact a fault tree can always be translated into an entirely equivalent set of Boolean equations. Thus an understanding of the rules of Boolean algebra contributes materially toward the construction and simplification of fault trees. Once a fault tree has been drawn, it can be evaluated to yield its qualitative and quantitative characteristics. These characteristics cannot be obtained from the fault tree per se, but they can be obtained from the equivalent Boolean equations.

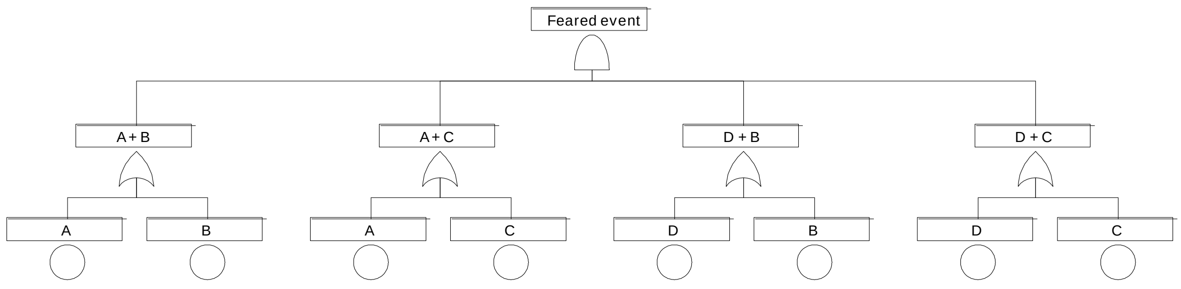
In this evaluation process, the following rules are used.

Designation	**Engineering symbolism**	**Comments**
Commutative law	$X \cdot Y = Y \cdot X$ $X + Y = Y + X$	/
Associative law	$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$ $X + (Y + Z) = (X + Y) + Z$	In the case of a series of "OR" operations or series of "AND" operations, the associative law permits to group the events as liked
Distributive law	$X \cdot (Y + Z) = X \cdot Y + X \cdot Z$ $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$	The distributive laws provide the valid manipulatory procedure whenever we have a combination of an "AND" operation with an "OR" operation
Idempotent law	$X \cdot X = X$ $X + X = X$	The idempotent laws allow to "cancel out" any redundancies of a same event.
Law of Absorption	$X \cdot (X + Y) = X$ $X + (X \cdot Y) = X$	/
Complementation	$X \cdot X' = \emptyset = 0$ $X + X' = \Omega = 1$ $(X')' = X$	/

Designation	Engineering symbolism	Comments
De Morgan's theorem	$(X \cdot Y)' = X' + Y'$ $(X + Y)' = X' \cdot Y'$	Suppose that X represents the failure of some component. Then X' represents the non-failure or successful operation of that component. In this light, the expression $(X \cdot Y)' = X' + Y'$ simply states that for the double failure of X and Y not to occur, either X must not fail or Y must not fail
Operations with 0 (Ø) and 1 (Ω)	$0 \cdot X = 0$ $0 + X = X$ $1 \cdot X = X$ $1 + X = 1$ $0' = 1$ $1' = 0$	/

Hereafter are given application examples for illustration of these rules.

<u>Example 1</u>: Simplification of an expression



(A + B) · (A + C) · (D + B) · (D + C) *(1)*

Distributive laws and law of absorption can be applied to (A + B) · (A + C) obtaining:

(A + B) · (A + C) = (A · A) + (A · B) + (A · C) + (B · C)

= A + (A · B) + (A · C) + (B · C)

= A + (B · C)

Likewise,

(D + B) · (D + C) = (D · D) + (D · B) + (D · C) + (B · C)

= D + (D · B) + (D · C) + (B · C)

= D + (B · C)

The expression (1) becomes:

(A + B) · (A + C) · (D + B) · (D + C) = [A + (B · C)] · [D + (B · C)] *(2)*

If E represents (B · C):

[A + (B · C)] · [D + (B · C)] = (A + E) · (D + E)

Distributive laws and law of absorption can be applied to (A + E) · (D + E) obtaining:

(A + E) · (D + E) = (A · D) + (A · E) + (D · E) + (E · E)

= E + (A · E) + (D · E) + (A · D)

= E + (A · D)

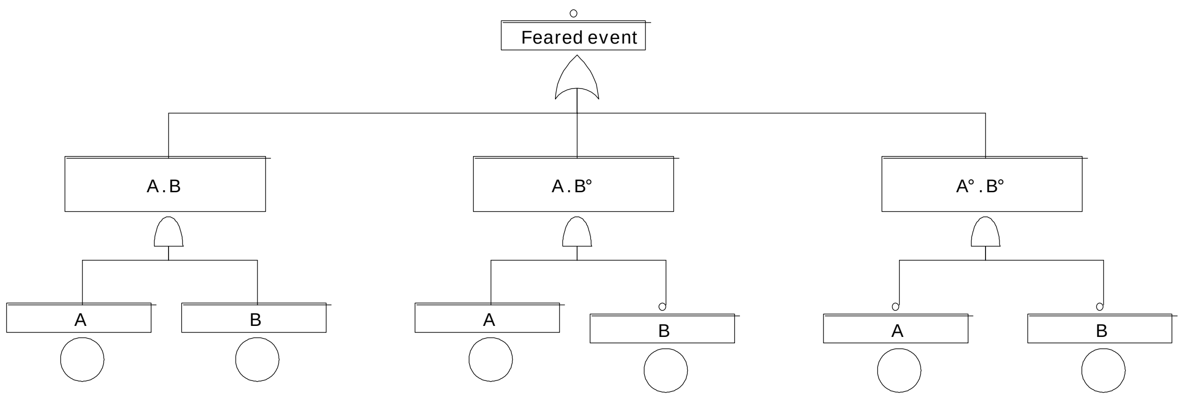
So the expression (2) becomes:

$$(B \cdot C) + (A \cdot D)$$

Therefore, the final result is:

$$(A + B) \cdot (A + C) \cdot (D + B) \cdot (D + C) = (B \cdot C) + (A \cdot D)$$

<u>Example 2</u>: Demonstration of the expression *(3)*



$[(A \cdot B) + (A \cdot B') + (A' \cdot B')]' = A' \cdot B$ *(3)*

This example can be worked either by **(a)** removing the outermost prime as a first step or by **(b)** manipulating the terms inside the large brackets and removing the outermost prime as a last step.

(a) $[(A \cdot B) + (A \cdot B') + (A' \cdot B')]'$

Using the de Morgan theorem:

$= (A \cdot B)' \cdot (A \cdot B')' \cdot (A' \cdot B')' = (A' + B') \cdot (A' + B) \cdot (A + B)$

Using the distributive law in the previous two first brackets:

$= [(A' \cdot A') + (A' \cdot B) + (A' \cdot B') + (B \cdot B')] \cdot (A + B)$

Using the idempotent law and law of Absorption:

$= [A' + (B \cdot B')] \cdot (A + B)$

Using the complementation and distributive laws:

$= (A' + \emptyset) \cdot (A + B) = A' \cdot (A + B) = (A' \cdot A) + (A' \cdot B) = \emptyset + (A' \cdot B)$

$= A' \cdot B$

(b) $[(A \cdot B) + (A \cdot B') + (A' \cdot B')]'$

Using the distributive law in the previous two first brackets:

$= [((A \cdot A) + (A \cdot B) + (A \cdot B') + (B + B')) + (A' \cdot B')]'$

Using the idempotent law and law of Absorption:

$= [((A \cdot (B + B')) + (A' \cdot B')]'$

Using the complementation law:

$$= [(A \cdot \Omega) + (A' \cdot B')]' = [A + (A' \cdot B')]'$$

By factorising the previous expression:

$$= [(A + A') \cdot (A + B')]'$$

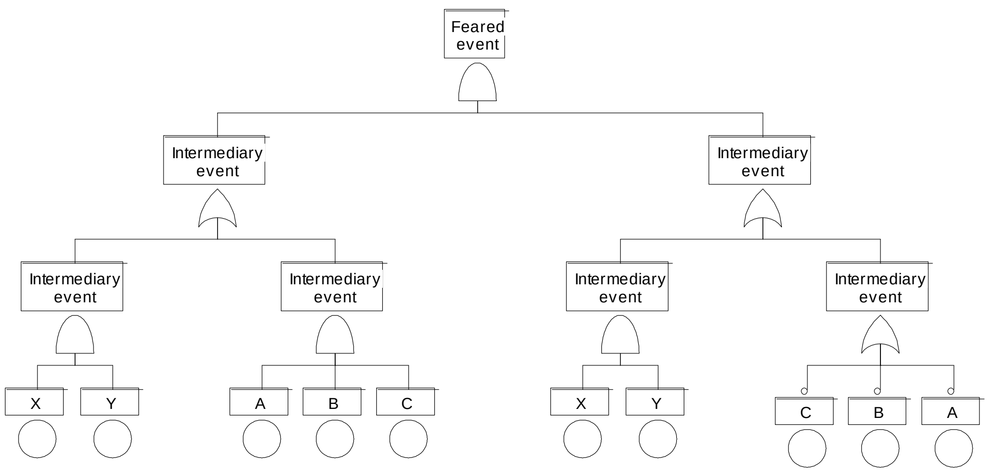
Using the complementation law:

$$= [\Omega \cdot (A + B')]' = [(A + B')]'$$

Using the de Morgan's theorem:

$$= A' \cdot B$$

<u>Example 3</u>: Demonstration of the expression *(4)*



$$[(X \cdot Y) + (A \cdot B \cdot C)] \cdot [(X \cdot Y) + (A' + B' + C')] \;=\; X \cdot Y \qquad (4)$$

De Morgan's theorem can be applied to the second term inside the second bracket:

$$[(X \cdot Y) + (A \cdot B \cdot C)] \cdot [(X \cdot Y) + (A \cdot B \cdot C)']$$

If $D = (X \cdot Y)$ and $E = (A \cdot B \cdot C)$, then:

$$[(X \cdot Y) + (A \cdot B \cdot C)] \cdot [(X \cdot Y) + (A \cdot B \cdot C)'] \;=\; (D + E) \cdot (D + E')$$

Using the distributive law:

$$= \; (D \cdot D) + (D \cdot E') + (D \cdot E) + (E \cdot E')$$

Using the idempotent, complementation and absorption laws:

$$= \; D + (D \cdot E') + (D \cdot E) + \emptyset$$

$$= \; D$$

$$= \; X \cdot Y$$

and the original statement is proved.

C.2 Minimal Cut Set

A minimal cut set is a smallest combination of component failures which, if they all occur, will cause the top event to occur.

By the definition, a minimal cut set is thus a combination (intersection) of primary events sufficient for the top event. The combination is a "smallest" combination in that all the failures are needed for the top event to occur. If any of the events in a minimal cut set does not occur, it prevents the top event from occurring (by this combination).

Any fault tree will consist of a finite number of minimal cut sets, which are unique for that top event. The one-event minimal cut sets, if there are any, represent those single events which will cause the top event to occur. The two-event minimal cut sets, if there are any, represent the doubles events which together will cause the top event to occur. For an n-event minimal cut set, all n events in the cut set must occur in order for the top event to occur.

The minimal cut set expression for the top event can be written in the following general form:

$$T=M_1 + M_2 + \ldots + M_i + \ldots + M_m$$

Where T is the top event and M_i are the minimal cut sets. Each minimal cut set consists of a combination of specific elementary events, and hence the general n-event minimal cut can be expressed as:

$$M_i = X_1 \cdot X_2 \ldots X_j \ldots X_n$$

where X_j is a basic event on the tree.

An example of a top event expression is:

$$T = A + (B \cdot C)$$

where A, B and C are elementary events.

This top event has one-event minimal cut set (A) and a two-event minimal cut set (B · C). The minimal cut sets are unique for a top event and are independent of the different equivalent forms the same fault tree may have.

To determine the minimal cut sets of a fault tree, the tree is first translated to its equivalent Boolean equations and then a substitution method is used.

The substitution method is straightforward and involves substituting and expanding Booleans expressions.

The redundancies are removed using the distributive and the absorption laws.

For small fault trees, the determination of the minimal cut sets can be done by hand. For larger trees, various computer algorithms and codes for fault tree evaluation are available.

In complex systems, the minimal cut set computation provides the analyst with a thorough and systematic method for identifying the basic combinations of events which can cause the undesired event (top event).

Hereafter are given application examples for illustration of these rules.

<u>Example 1</u>: Simple fault tree

Considering the following fault tree (Figure 5.2), the Boolean equations are shown hereafter.

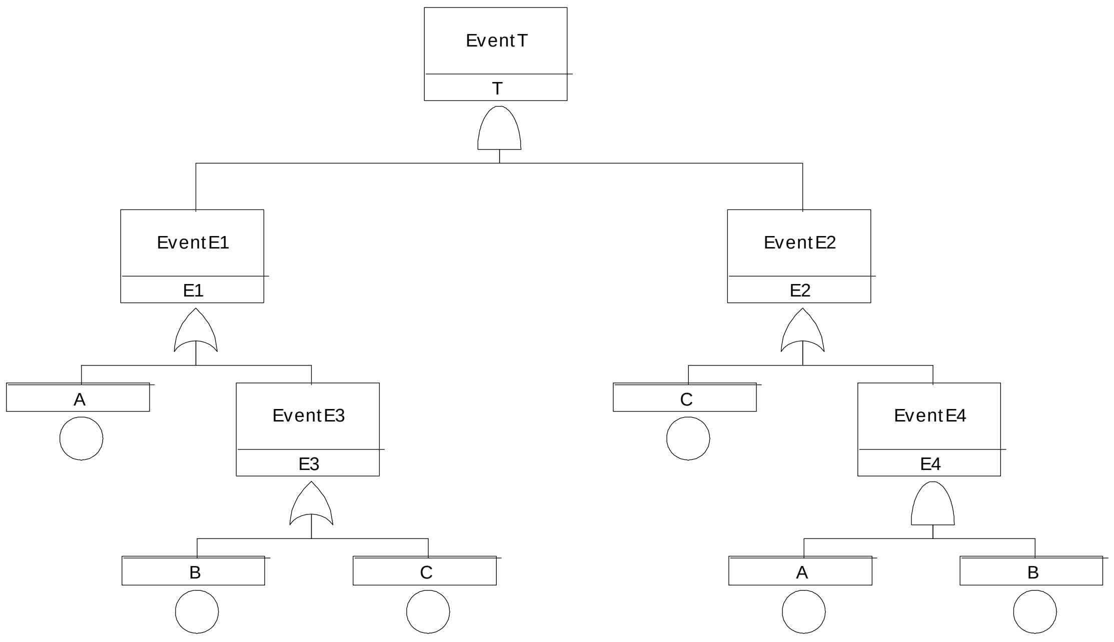


Figure 5.2: Simple fault tree

The tree is first translated to its equivalent Boolean equations

$T = E1 \cdot E2$

$E1 = A + E3$

$E3 = B + C$

$E2 = C + E4$

$E4 = A \cdot B$

The substitution is then done.

Substituting for E1 and E2 and expanding:

$$T = (A + E3) \cdot (C + E4)$$

$$= (A \cdot C) + (A \cdot E4) + (E3 \cdot C) + (E3 \cdot E4)$$

Substituting for E3:

$$T = (A \cdot C) + (A \cdot E4) + ((B + C) \cdot C) + ((B + C) \cdot E4)$$

$$= (A \cdot C) + (A \cdot E4) + (B \cdot C) + (C \cdot C) + (B \cdot E4) + (C \cdot E4)$$

Using the idempotent law ($C \cdot C = C$) and the absorption law ($C + x{\cdot}C = C$):

$$T = (A \cdot E4) + C + (B \cdot E4)$$

Substituting for E4:

$$T = (A \cdot (A \cdot B)) + C + (B \cdot (A \cdot B))$$

$$= (A \cdot A \cdot B) + C + (B \cdot A \cdot B)$$

Using the idempotent law ($A \cdot A = A$ and $B \cdot B = B$ and $A \cdot B + A \cdot B = A \cdot B$):

$$T = C + (A \cdot B)$$

The minimal cut sets of the top event are thus C and ($A \cdot B$): one single event minimal cut set and one double event minimal cut set.

The fault tree can thus be represented as shown in the Figure 5.3, which is equivalent to the original tree (both trees have the same minimal cut sets)

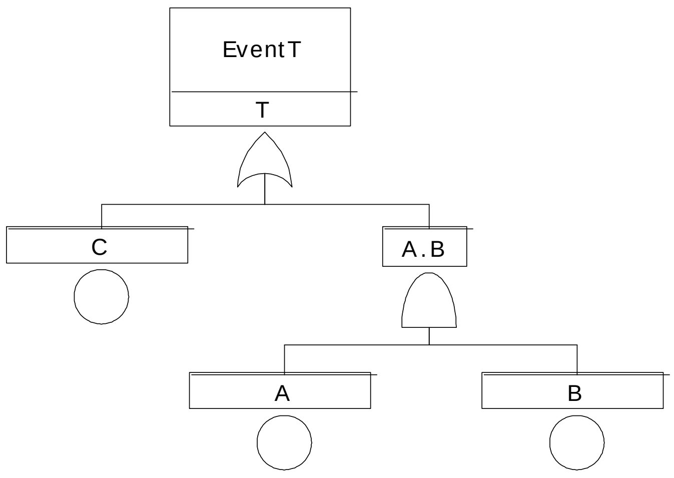


Figure 5.3: Representation of the minimal cut sets

<u>Example 2</u>: "Complex" fault tree

Considering the following fault tree (Figure 5.4), the Boolean equations are shown hereafter.

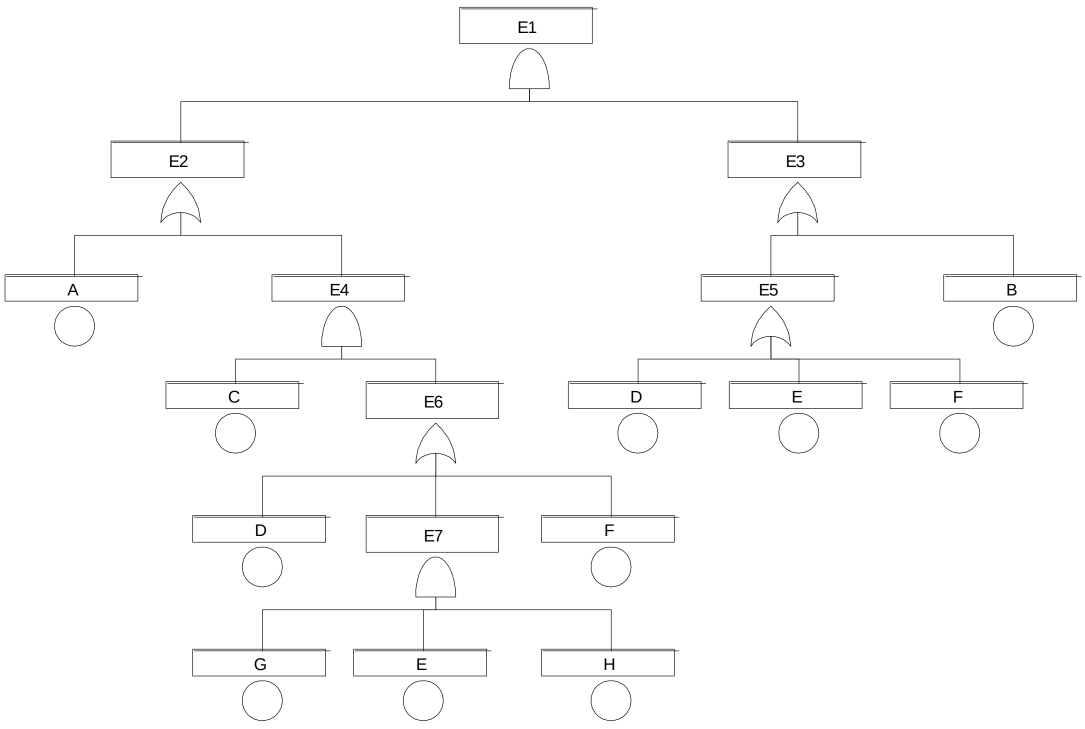


Figure 5.4: Complex fault tree

The tree is first translated to its equivalent Boolean equations

$E1 = E2 \cdot E3$

$E2 = A + E4$

$E3 = B + E5$

$E4 = C \cdot E6$

$E5 = D + E + F$

$E6 = D + E7 + F$

$E7 = E \cdot G \cdot H$

The substitution is then done.

$E1 = (A + E4) \cdot (B + E5)$

$E1 = (A \cdot B) + (A \cdot E5) + (E4 \cdot B) + (E4 \cdot E5)$

$E1 = (A \cdot B) + (A \cdot (D + E + F)) + (C \cdot E6 \cdot B) + (C \cdot E6 \cdot (D + E + F))$

$$E1 = (A \cdot B) + (A \cdot D) + (A \cdot E) + (A \cdot F) + (C \cdot (D + E7 + F) \cdot B) + (C \cdot (D + E7 + F) \cdot (D + E + F))$$

$$E1 = (A \cdot B) + (A \cdot D) + (A \cdot E) + (A \cdot F) + (C \cdot (D + E7 + F) \cdot B) + (C \cdot ((D \cdot D) + (D \cdot E) + (D \cdot F) + (E7 \cdot D) + (E7 \cdot E) + (E7 \cdot F) + (F \cdot D) + (F \cdot E) + (F \cdot F)))$$

$$E1 = (A \cdot B) + (A \cdot D) + (A \cdot E) + (A \cdot F) + (C \cdot (D + E7 + F) \cdot B) + (C \cdot (D + (E7 \cdot E) + F))$$

$$E1 = (A \cdot B) + (A \cdot D) + (A \cdot E) + (A \cdot F) + (C \cdot (D + E7 + F) \cdot B) + (C \cdot (D + (E \cdot G \cdot H \cdot E) + F))$$

$$E1 = (A \cdot B) + (A \cdot D) + (A \cdot E) + (A \cdot F) + (C \cdot B \cdot D) + (C \cdot B \cdot E7) + (C \cdot (D + (E \cdot G \cdot H) + F))$$

$$E1 = (A \cdot B) + (A \cdot D) + (A \cdot E) + (A \cdot F) + (C \cdot B \cdot D) + (C \cdot B \cdot E7) + (C \cdot D) + (C \cdot E \cdot G \cdot H) + (C \cdot F)$$

$$E1 = (A \cdot B) + (A \cdot D) + (A \cdot E) + (A \cdot F) + (C \cdot B \cdot E \cdot G \cdot H) + (C \cdot D) + (C \cdot E \cdot G \cdot H) + (C \cdot F)$$

The result is:

$$E1 = (A \cdot B) + (A \cdot D) + (A \cdot E) + (A \cdot F) + (C \cdot D) + (C \cdot F) + (C \cdot E \cdot G \cdot H)$$

The minimal cut sets are:

$A \cdot B$

$A \cdot D$

$A \cdot E$

$A \cdot F$

$C \cdot D$

$C \cdot F$

$C \cdot E \cdot G \cdot H$

C.3 *Qualitative approach*

Qualitative approach includes:

- the minimal cut sets of the fault tree;
- the qualitative importance;
- the common cause events.

Minimal cut sets

As previously discussed, the minimal cut sets give all the unique combinations of basic events that cause the top event.

Qualitative importances

The qualitative importances give a "qualitative ranking" on each basic event with regard to its contribution to the top event.

After obtaining the minimal cut sets, some idea of basic event importances can be obtained by ordering the minimal cut sets according to their size. The single-event minimal cut sets (if any) are listed first, then the double-event minimal cut sets, then the triple, etc.

It is often the practice to sort only the single, double, and perhaps triple-event minimal cut sets. As an additional calculation, higher order minimal cut sets (quadruples, etc.) can also be sorted if they show potential susceptibility to common cause events.

Because the basic event probabilities associated with the minimal cut sets often decrease by orders of magnitude as the size of the cut set increases, the ranking according to size gives a gross indication of the importance of the minimal cut set. For example, if individual basic event probabilities are of the order of 10^{-3}, a single-event cut set probability will be of the order of 10^{-3}, and a double cut set 10^{-6}, a triple 10^{-9},etc.

Basic event probabilities are in general different, therefore the ranking of minimal cut sets according to size gives only a general indication of importance.

The minimal cut set information can sometimes be used directly to check design criteria. For example, if a design criterion states that no single event shall lead to the top event, then this is equivalent to stating that the system shall contain no single event minimal cut sets. The minimal cut sets can be checked to see if this criterion is satisfied.

Common cause susceptibilities

In evaluating a fault tree, the events which could be common cause events are not known. However, the susceptibility that component failures may have a common initiating cause can be indicated.

The basic events on a fault tree do not necessarily have to be independent. A single, more basic cause may result in multiples events which lead to the top event. Multiple events which can end in the top event and which can originate from a common cause are termed "common cause events".

By definition, the top event occurs if all the basic events in a minimal cut set occur. So, what is interesting are only the common causes which can trigger all the basic events in a minimal cut set. A cause which does not trigger all the basic events in a minimal cut set will not by itself cause system failure.

To identify minimal cut sets which are susceptible to common cause events, common cause categories can be defined. These are general areas that can cause event dependence.

The list below gives some example categories which might be considered in a common cause susceptibility evaluation:

- Location;
- Temperature;
- Humidity;
- Flood susceptibility;
- Manufacturer;
- Wear-out susceptibility;
- Maintenance degradation;
- Operator interactions (human factor);
- Energy sources;
- Etc.

For each common cause category, specific "elements" can be defined. For example, for the category "Manufacturer" the elements would be particular manufacturers involved which might be coded as "Manufacturer 1", "Manufacturer 2", etc. For the "Location" category, a system can be divided into a given number of physical locations which would be the elements. For the category "Flood susceptibility", several sensitivity levels might be defined ranging from no sensitivity to extreme sensitivity.

The next task in the common cause susceptibility evaluations involves event coding. As part of the event name code (reference) or in associated event description fields, for each event occurrence, the element of each category associated with the basic event can be denoted. The categories can be indexed or keyed according to any convenient coding system.

Having performed this coding, the potentially susceptible minimal cut sets can be identified among the collection of minimal cut sets determined for the fault tree. The minimal cut sets which are potentially susceptible to common cause failures are those whose primary failures all have the same element of a given category. Having identified the potentially susceptible minimal cut sets, the minimal cut sets need finally to be screened in order to determine those which may require further action. This final screening may be based on past histories of common cause occurrences, some sort of quantification analysis, and/or expert judgement.

D

APPENDIX D QUANTITATIVE APPROACH

The purpose of quantitative (numerical) evaluation is to provide a quantitative assessment of the safety measure (e.g. probability of occurrence) of the top event or of a selected set of events. Quantitative evaluation is also used to supplement the qualitative evaluation.

In addition to the evaluation of the safety measure(s) associated to the top event (or a selected event), the quantitative analysis encompasses the importance analysis, the sensitivity analysis and the uncertainty analysis.

D.1 Importance analysis

An importance analysis is used for evaluating the role of an elementary event in the occurrence of the top event [7].

Importance factors seek to answer the question:

- "What are the most important contributors to the safety measure associated to the top event?"

in order to identify the most efficient strategy of improving the system safety by improving the probabilistic characteristics of its elements.

Several importance factors exist:

- Marginal Importance Factor: Also called "Birnbaum importance factor", it was introduced by Birnbaum [1]. It is the rate at which the availability of the system (top event) increases when the availability of the "component" (basic event) increases.
  The marginal importance factor may also be interpreted as the probability of system S being in a state such that if, all other things being equal, the component e has failed, then the system fails, otherwise it works.

- Critical Importance Factor: This importance factor, introduced by Lambert [2] [3], is for evaluating the relative importance of the components (basic events). It is, by definition, highly dependent on the marginal importance factor.
  It is the probability that a basic event causes the top event (system failure), knowing that the top event has occurred.

  The critical importance factor can be calculated for a minimal cut set as well. This is the probability that the minimal cut set causes the top event knowing that the top event has occurred. The critical importance factor is really interesting because it gives the respective weight of each minimal cut in the contribution to the system failure.

- Diagnostic Importance Factor (Vesely - Fussel): Introduced by Vesely and Fussel [4], it is the probability of one basic event having caused

the top event, given that the top event has effectively occurred.
It is useful for diagnosing the causes for the top event occurrence.

- Risk Increase Factor: It is an indicator of the importance of maintaining the level of reliability of a component in maintaining the reliability of the system [5].

  It should be interpreted with extreme caution as it is only a rough approximation [6].

- Risk Decrease Factor: It is the maximum decrease in risk which can be expected by improving the reliability of a component. It is important for identifying those components whose improved reliability is most likely to increase that of the system.

D.2 Sensitivity analysis

A sensitivity analysis is used for observing the variation on the probability of the top event, induced by the variation of probabilistic parameters assigned to the basic events.

The variation of probabilistic parameters describes aspects like changing maintenance and checking times, implementing design modifications or changing component reliabilities. The result is a fine evaluation of the contribution of those parameters to the safety measure evaluated for the top event.

It is particularly convenient to assess effects of component data variations using the formulas presented in the section "Quantitative evaluation" as they explicitly contain component failure rates, repair times and test intervals as variables. For example, if T_D are mean down times for the components of a radio antenna (including the repair time plus all the logistic times related to the access to the remote site, the time to get the spare components, etc), then the effects on that antenna unavailability with regard to different times for access to the remote site can be studied by varying the T_D s for those components accordingly.

Scoping-type evaluations can also be performed by using, as an example, a high failure rate and a low failure rate for a particular basic event in the tree. If the system unavailability doesn't change significantly, then the event is not important and no more attention needs to be paid on it. If the system unavailability does change significantly, then more accurate probabilistic data must be obtained for that particular component, or the event must be further developed to more basic causes.

A wide spectrum of sensitivity analysis can be performed, depending on the needs of the safety engineer.

NOTE: Provided that adequate SW tools are available for supporting it, the sensitivity analysis should be preferred to the analysis based on importance factors.

D.3 Uncertainty analysis

An uncertainty analysis aims at estimating the degree of uncertainty of the safety measure associated to the top event, based on assumptions on the uncertainty associated to the probabilistic data assigned to the basic events.

In judging the significance of an effect, it is important that the safety analyst take into account the precision of the data input to the Fault Tree quantification.

Software tools do that functionality by assigning a distribution (i.e. lognormal distribution) to a parameter of a basic event (i.e. probability) and then by running Monte Carlo simulation on the calculation of the probability of the top event.

D.4 Bibliography

[1] Z. W. Birnbaum – *On the importance of different components and a multicomponent system* – Multivariate Analysis II – P. R. Krishnaiah Editor, 1969

[2] H. E. Lambert – *Fault trees for decision making in system analysis* – Ph. D. Thesis, Lawrence Livermore Laboratory, 1975

[3] H. E. Lambert – *Measures of importance of events and cut sets in fault trees* – In R.E. Barlow, J.B. Fussel and N.D. Singpurwalla, editors, Reliability and Fault Tree Analysis, pages 77 to 100 – SIAM Press, 1975

[4] J. B. Fussel – *How to hand-calculate system reliability characteristics* – IEEE Trans. On Reliability, R-24 (3), 1975

[5] U. Berg – *RISK SPECTRUM, Theory Manual* – RELCON Teknik AB, April 1994

[6] I.B. Wall and D.H. Worledge – *Some perspectives on risk importance measures. In Proceedings of the international conference on Probabilistic Safety Assessment* – PSA 96, 1996

[7] A. Villemeur – *Reliability, availability, maintainability and safety assessment* – John Wiley and Sons, Inc., 1991

E

APPENDIX E QUANTITATIVE ALLOCATION THROUGH THE FAULT TREE

AND and OR logical gates

According to the fault tree, safety objectives are allocated to the different elementary causes in order to derive quantitative probability requirements that prevent the hazard from occurring with a probability higher than the one allocated by the corresponding Safety Objective.

As for the qualitative allocation method, two apportionment strategies could be used for quantitative apportionment:

- A "top-down" strategy, which aims at apportioning the high level quantitative Safety Objective into low level Safety Requirements, accounting for the data available, which represent the constraints;
- A "bottom-up" strategy, which aims at filling the missing safety data in the fault tree (by "guessing" the Safety Requirements to be allocated to the elementary events) in order to reach the qualitative Safety Objective at the top of the tree.

Another strategy can be used to apportion Safety Objectives into Safety Requirements. It is based on the Minimal Cut Set. This method permits to take into account the possible common events.

NOTES:

- **The basic events are considered independent.**
- In the fault tree, when the breakdown is represented as an OR relationship then this means that any of the lower paths could independently cause the undesired event to occur. Probabilities of the lower leafs would normally sum to achieve the upper level.
- An AND relationship means that all the conditions must be present at the same time in order to create the hazard. In such a case, there are generally several possibilities to allocate the probabilities of the lower leafs (probabilities "multiply").
- No common event is considered: if some common basic event exists, the allocation is to be made using the minimal cut sets (see explanations in the chapter **Error! Reference source not found.**).

ABOUT THE PROBABILITIES:

Each probability can be considered as a scientific number "X. 10^{-y}" or "X E^{-y}".

The number X, member of the probability assigned to each input, could include several decimal places. To simplify the apportionment process, each probability can be considered as a scientific number "X.10^{-y}" or "X E^{-y}" where X is an integer (whole number).

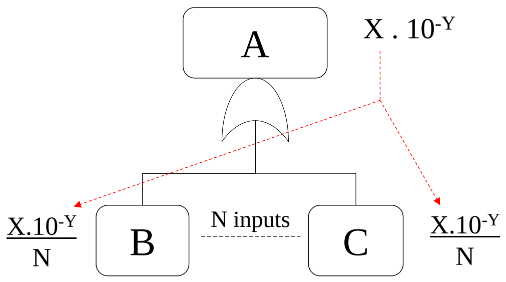
The process to convert a decimal number into an integer must still be "safety conservative". The number X must be rounded down to the nearest integer.

E.1 "Top-down" strategy

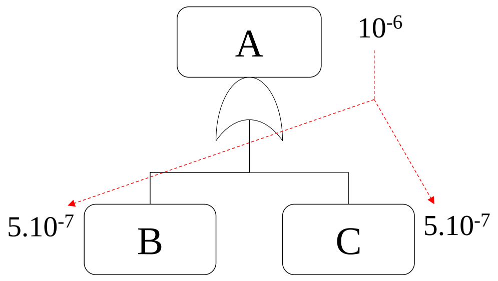
The safety objective allocation could be based on the following principles:

- <u>from an OR gate</u>, two allocation methods are possible:

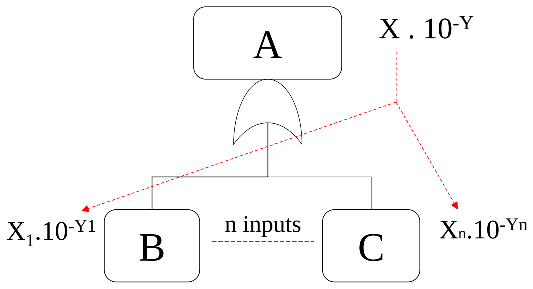
  1. The safety objective can be apportioned in an equiprobable manner to each input event, as shown hereafter.



For example:



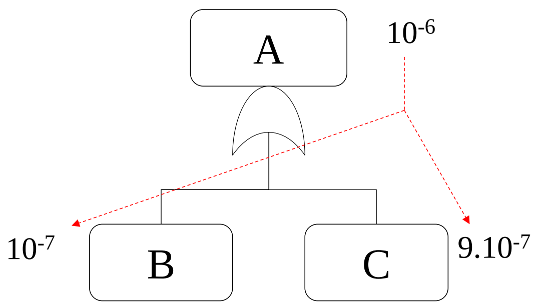
  2. The safety objective can be apportioned with no equiprobable consideration to each input event, as shown hereafter.

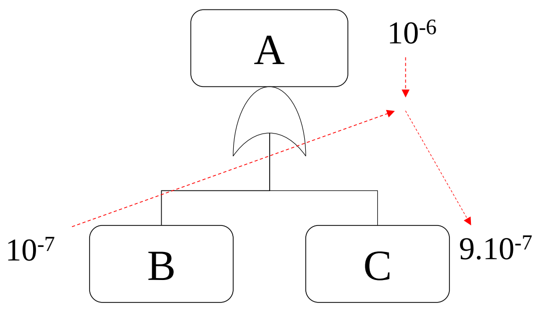


<u>With</u>: $X.10^{-Y} \geq X_1.10^{-Y1} + \ldots + Xn.10^{-Yn}$

It is dependant on experience feedback or field experience, similar component failures or human errors addressed by industry reliability data bases, state-of-the art for ensuring a certain level of reliability/safety for a certain type of system element, etc.

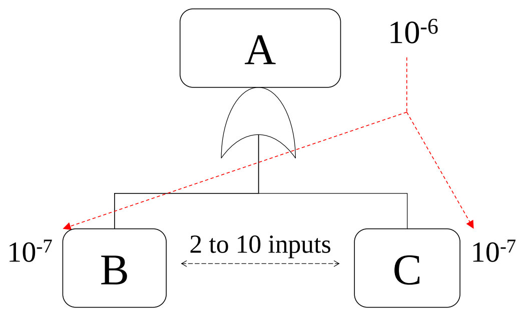
For example:

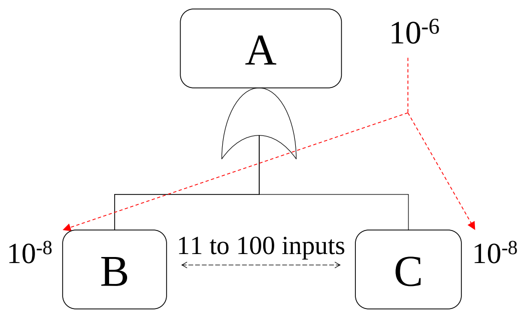




For the two previous proposed methods, what is important is that the combination of all inputs would be a conservative "orders of magnitude" of the output objective.

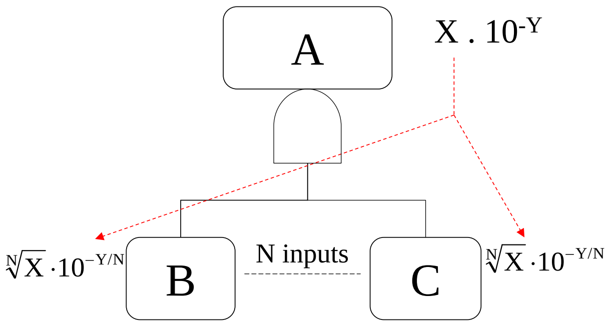
<u>NOTE</u>: Most of the time, the apportionment probabilities are rounded down to the nearest ten. If the OR gate safety objective is to be apportioned to 2 to 10 gate inputs, then the probability for an input is the output probability divided by ten. If the OR gate safety objective is to be apportioned to 11 to 100 gate inputs, then the probability for an input is the output probability divided by one hundred. If the OR gate has only one input, the objective is the same.





- from an AND gate, two allocation methods are possible:

    1. The safety objective can be apportioned in an equiprobable manner to each input event, as shown hereafter.

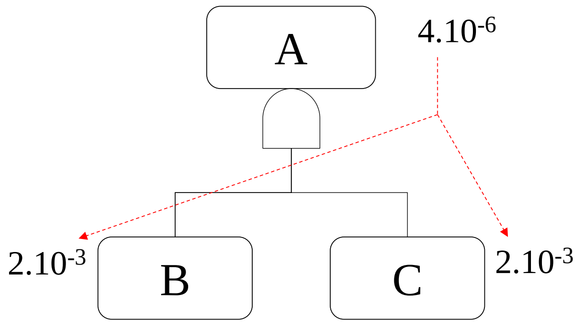


The equiprobable apportionment of an AND gate objective depending on the number N of inputs is the following (considering that the N inputs are independent):

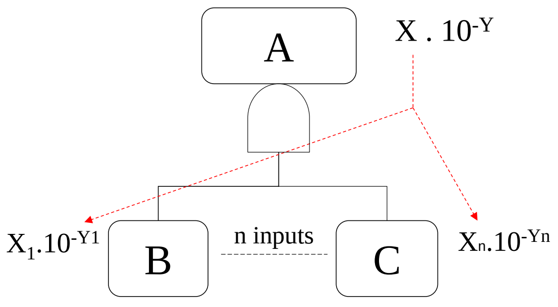
If the output objective is **X . 10^{-Y}** then for one input the objective is:

$$\sqrt[N]{X \cdot 10^{-Y}} = \sqrt[N]{X} \cdot 10^{-Y/N}$$

For example:



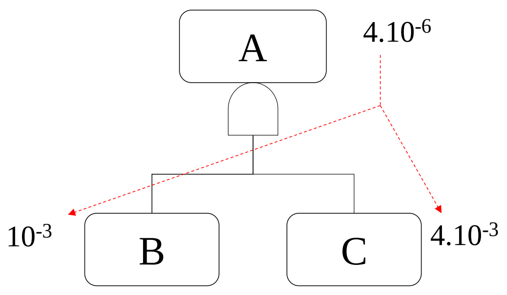
2. The safety objective can be apportioned with no equiprobable consideration to each input event, as shown hereafter.

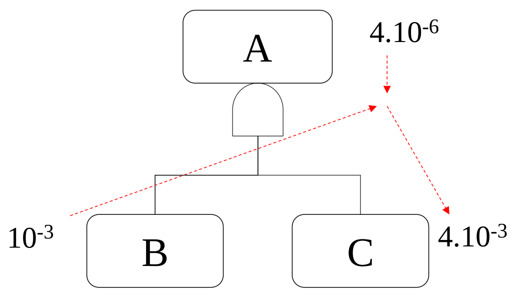


With: $X.10^{-Y} \geq X_1.10^{-Y1}$. … . $Xn.10^{-Yn}$

It is dependant on experience feedback or field experience, similar component failures or human errors addressed by industry reliability data bases, state-of-the art for ensuring a certain level of reliability/safety for a certain type of system element, etc.

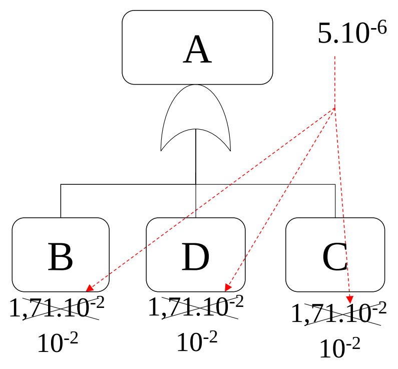
For example:





What is important is that the combination of all inputs would be a conservative "orders of magnitude" of the output objective.

NOTE: Most of the time, the apportionment probabilities are rounded down to the nearest ten. For example:

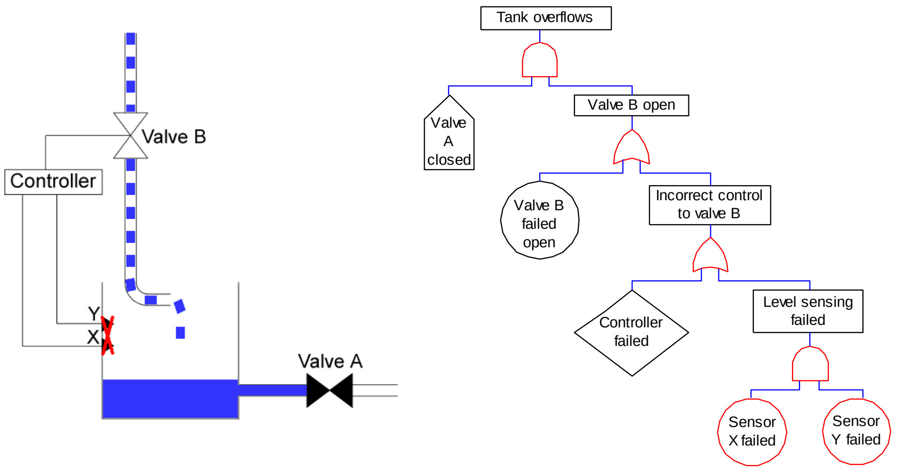


As the presented propositions are only guidelines, SO allocated through an OR or AND gate can be different as here proposed, depending on the expert judgement and other factors. In consequent, an explanation of the allocation through each gate needs to be added after each allocation fault tree, when the choice/decision cannot be merely explained in an "assumptions/rules" paragraph or table.

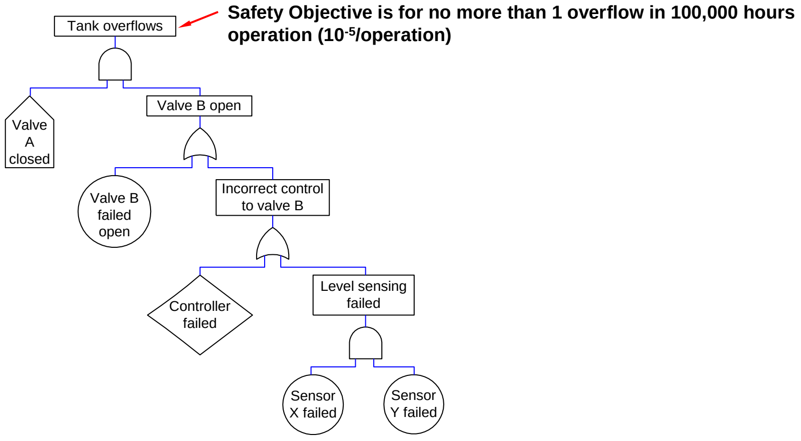
E.2 *“Top-Down” Strategy Example*

The following simple example illustrates the application of the “Top-Down” strategy.

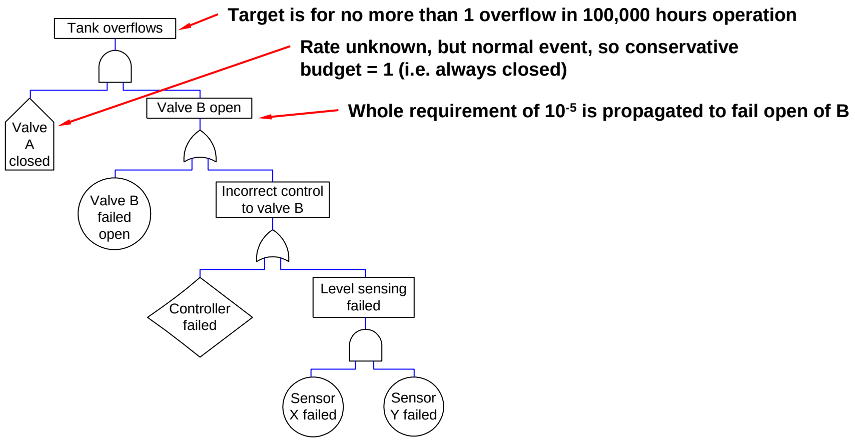
Simple Example: Tank Overflows



Use of FTA in PSSA



Use of FTA in PSSA

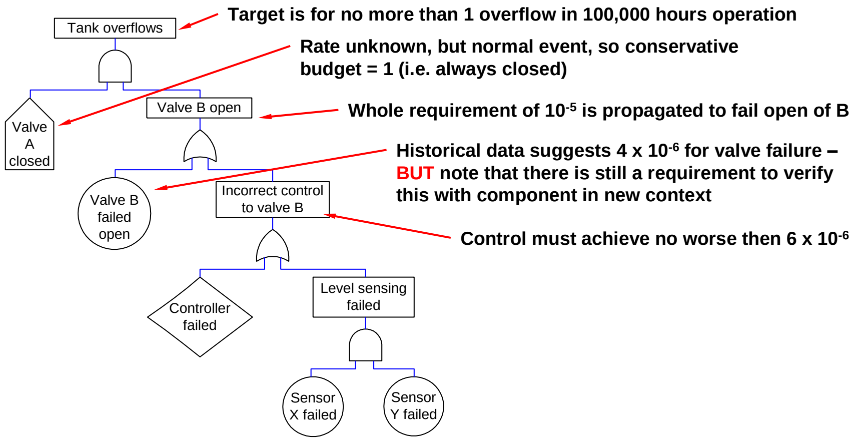


Target is for no more than 1 overflow in 100,000 hours operation

Rate unknown, but normal event, so conservative budget = 1 (i.e. always closed)

Whole requirement of 10^{-5} is propagated to fail open of B

Use of FTA in PSSA



Target is for no more than 1 overflow in 100,000 hours operation

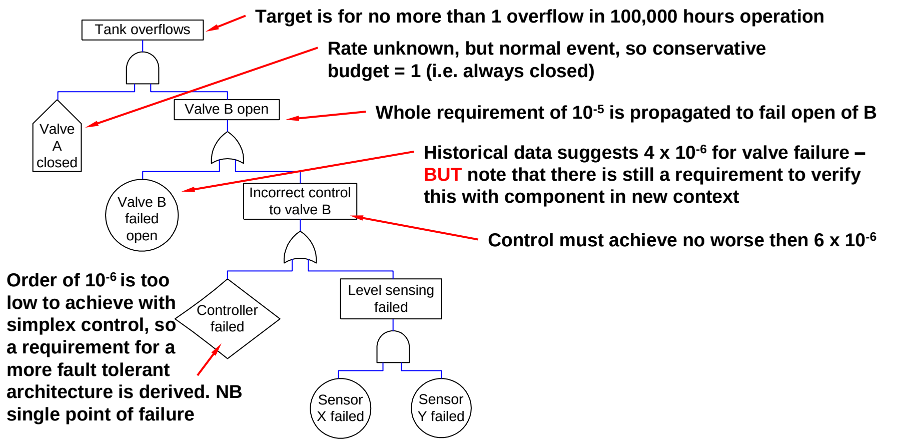
Rate unknown, but normal event, so conservative budget = 1 (i.e. always closed)

Whole requirement of 10^{-5} is propagated to fail open of B

Historical data suggests 4 x 10^{-6} for valve failure – BUT note that there is still a requirement to verify this with component in new context

Control must achieve no worse then 6 x 10^{-6}

Use of FTA in PSSA



Target is for no more than 1 overflow in 100,000 hours operation

Rate unknown, but normal event, so conservative budget = 1 (i.e. always closed)

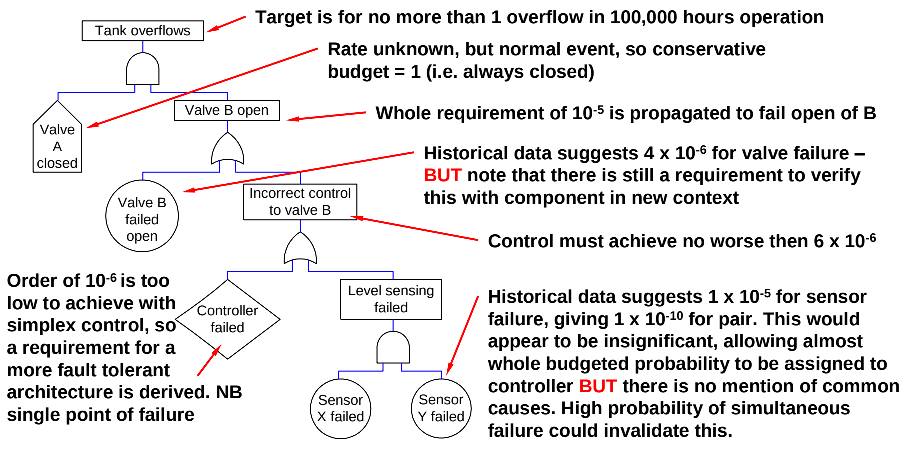
Whole requirement of 10^{-5} is propagated to fail open of B

Historical data suggests 4 x 10^{-6} for valve failure – BUT note that there is still a requirement to verify this with component in new context

Control must achieve no worse then 6 x 10^{-6}

Order of 10^{-6} is too low to achieve with simplex control, so a requirement for a more fault tolerant architecture is derived. NB single point of failure

Use of FTA in PSSA



Target is for no more than 1 overflow in 100,000 hours operation

Rate unknown, but normal event, so conservative budget = 1 (i.e. always closed)

Whole requirement of 10^{-5} is propagated to fail open of B

Historical data suggests 4 x 10^{-6} for valve failure – BUT note that there is still a requirement to verify this with component in new context

Control must achieve no worse then 6 x 10^{-6}

Order of 10^{-6} is too low to achieve with simplex control, so a requirement for a more fault tolerant architecture is derived. NB single point of failure

Historical data suggests 1 x 10^{-5} for sensor failure, giving 1 x 10^{-10} for pair. This would appear to be insignificant, allowing almost whole budgeted probability to be assigned to controller BUT there is no mention of common causes. High probability of simultaneous failure could invalidate this.

Therefore examples of Safety Requirements derived from the Safety Objective (The frequency that the tank overflows shall be no greater than once per 100.000 operations (or 10^{-5}/operation)) that can be allocated to these elements are:

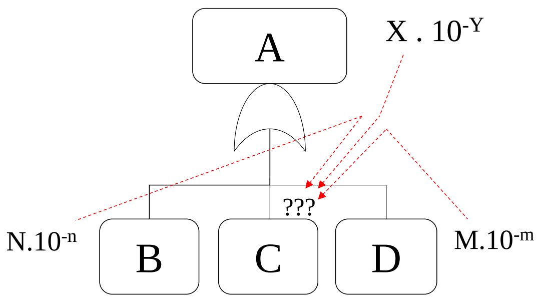
- Valve B
  - SR-VB-1: Valve B shall "Fail open" no greater than 4 x10^{-6} per operating hour.
- Controller
  - SR-C-1: Controller shall not fail to close valve B more than 6 x10^{-6} per operating hour.
  - SR-C-2: Redundancy shall be used to meet the above failure rate requirement.
  - SR-C-3: Controller Software shall demonstrate a SWAL3.
- Sensors
  - SR-S-1: Sensors shall not fail to detect presence of liquid more than 10^{-5} per operating hour.
  - SR-S-2: Demonstration of independence (dissimilarity) shall be made during installation as well as in the design.
  - SR-S-3: If Sensors are not independent, the failure rate of this equipment shall be revisited (be more stringent).

E.3 *"Bottom-Up" strategy*

The safety objective allocation could be based on the following principles:

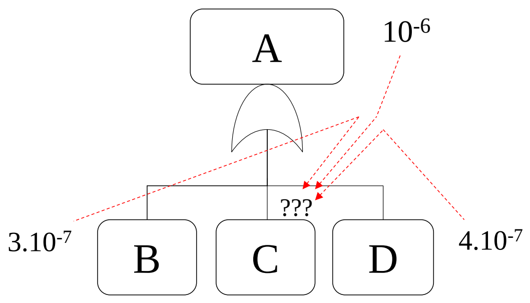
- <u>from an OR gate</u>:

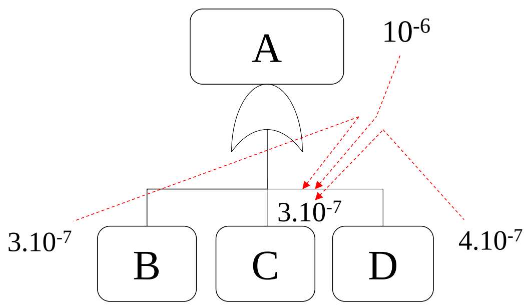
The difference between the Safety Objective and the sum of all the known quantitative requirements is determined.



Then, this difference is apportioned (even or not) to the basic events for which no safety requirement was defined.

For example:

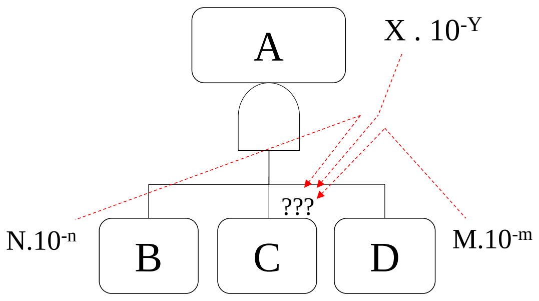




Note that for this method, the Safety Objective must be greater than the sum of all the known quantitative requirements. If not, the apportionment cannot be done because the result of the fault tree is greater than the defined Safety Objective.

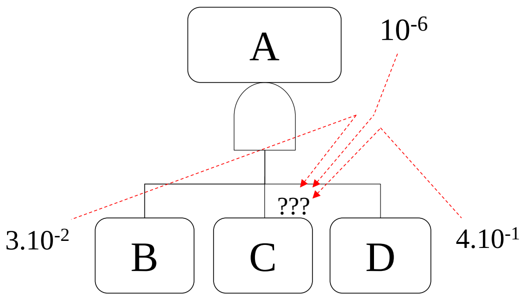
- from an AND gate:

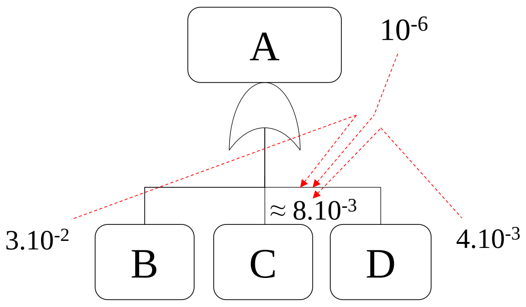
The Safety Objective is divided by the product of all the known quantitative requirements.



Then, the result of the division is apportioned (the product of all the apportioned Safety Requirements, in an even manner (using an "nth root") or not, must be less than the result of the division) to the basic events for which no safety requirement was defined.

For example:





Note that if the product of all the known quantitative requirements is less than the required Safety Objective, then the basic events, for which a quantitative requirement is needed, can be allocated with a probability of 1.

E.4 "Minimal Cut Set" strategy

The "Minimal Cut Set" strategy can be used either the fault tree includes common basic events or not.

Because of the definition of the minimal cut sets, the apportionment of Safety Objectives into Safety Requirements using this method requires to apportion the objectives through AND and OR gates.

Indeed, as seen in the APPENDIX C, the minimal cut set expression for the top event can be written in the following general form:

$$T=M_1 + M_2 + \ldots + M_i + \ldots + M_m$$

Where T is the top event and M_i are the minimal cut sets. Each minimal cut set consists of a combination of specific elementary events, and hence the general n-event minimal cut can be expressed as:

$$M_i = X_1 \cdot X_2 \ldots X_j \ldots X_n$$

Where X_j is a basic event on the tree.

The "Minimal Cut Set" method can be used to realize the apportionment of a Safety Objective knowing or not Safety Requirements.

If no Safety Requirement is known because no experience feed-back exists for example: the same rules that the ones defined in the chapter "Top-down" strategy can be applied.

Firstly, the Safety Objective is propagated through the OR gate ($T=M_1 + M_2 + \ldots + M_i + \ldots + M_m$).

Then, for each minimal cut, the apportionment is done through an AND gate ($M_i = X_1 \cdot X_2 \ldots X_j \ldots X_n$).

If some Safety Requirements are known (experience feed-back, expert judgment, etc.) the missing safety data can be filled in the minimal cut sets (by "guessing" the Safety Requirements to be allocated to the elementary events). It uses the same rules than the ones defined in the chapter “Bottom-Up” Strategy.

It could be processed with iteration in order to reach the qualitative Safety Objective at the top of the tree.