ExpertFTA: An Expert's Knowledge Based Software Tool for Fault Tree Analysis

BY

Syed Mahmudul Hasan

A thesis submitted to the School of Graduate Studies
in partial fulfillment of the requirements for the degree of

MASTER OF COMPUTATIONAL SCIENCE

Supervisor: Dr. Faisal Khan

DEPARTMENT OF COMPUTATIONAL SCIENCE

MEMORIAL UNIVERSITY OF NEWFOUNDLAND

St. John's, Newfoundland, Canada

July, 2012

Abstract

Having an effective and systematic risk analysis and safety management strategy is imperative to avoid unwanted accidents in any process facility. Fault Tree Analysis (**FTA**) is a frequently used technique by design engineers for Probabilistic Risk Assessment (PRA). Imprecise, incomplete and vagueness of data can result uncertainty in output from FTA. The Dempster–Shafer Theory of Evidence (**DST**) addresses the incompleteness in data while fuzzy theory handles impreciseness or vagueness in data.

A computer aided tool for FTA, ***ExpertFTA-*** is introduced in this thesis. Both **DST** and fuzzy theory are considered to develop this software tool in order to aggregate knowledge from multiple experts. ***ExpertFTA*** can assists users (with little knowledge of FTA) to draw a fault tree and perform the analysis effectively. ***ExpertFTA*** helps users to create a fault tree, modify it and store (profiling) data for future reference. Users can perform qualitative, quantitative and sensitivity analysis of the fault tree from DST and fuzzy point of view. It also provides a report based on the generated fault tree. Several established design patterns are implemented and object oriented concepts of Java, XML and XSLT are used in the development of ***ExpertFTA***.

None of the currently available commercial software for FTA has the capability of performing analysis based on DST and fuzzy logic. This tool is developed with the anticipation of using it for research purposes and also for industry personnel for detailed risk analysis. It is designed such a way that it can be extended with more functionality in the future.

Acknowledgements

First of all, I would like to express my deepest gratitude to Almighty Allah, the most gracious, the most merciful who has given me the strength and ability to finish this thesis.

I am truly grateful and would like to give my cordial thanks to my supervisor Dr. Faisal Khan for his support and guidance. I would like to express my sincere appreciation to Dr. Refaul Ferdous, who has tremendously helped me whenever needed. I am also greatly indebted to the Department of Computational Science for giving me the opportunity to pursue my Master's degree and to Jason Mills for proofreading my thesis.

I could not thank enough to my wife Syeda Tanzila Kamal for her unconditional and unbound love, support and inspiration to pursue my Masters degree.

Last but not least, I am deeply thankful to my parents and sister for their remarkable supports in all aspects. And I would like to dedicate this thesis to my father Syed Tofazzal Hossian who was the source of continuous inspiration to pursue this degree.

Glossary

List of Symbols

μ	Membership function of a fuzzy set
Ω	Frame of Discernment
P	Power set
p_i	Subset of power set
P_{OR}	"OR" gate operation
P_{AND}	"AND" gate operation
Φ	Null set
BE_i	Basic Events as input events
$m(p_i)$	Belief mass or basic probability assignment
k	Degree of conflict
p_L	Lower boundary
p_m	Most likely value
p_U	Upper boundary
p_f	Degree of membership
$\widetilde{P}_{\alpha}$	Alpha-cut
β	Probability value
α	Fuzzy Number

List of Abbreviations

ExpertFTA	Expert knowledge based Fault Tree Analysis
AIChE	American Institute of Chemical Engineers
BE, GE, CE, TE	Basic Event, Gate Event, Condition Event, Transfer Event
FTA	Fault Tree Analysis
HAZOP	Hazard and Operability
PRA	Probabilistic Risk Assessment
PROFAT	PRObabilistic FAult Tree
FOD	Frame of discernment
ETA	Even Tree Analysis
TFN	Triangular Fuzzy Number
ZFN	Trapezoidal Fuzzy Number
Bel,Pl	Belief, Plausibility
T, F	True, False
DST	Dempster–Shafer Theory of Evidence
bpa	basic probability assignment
OOP	Object Oriented Programming
GUI	Graphical User Interface
UML	Unified Modeling Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations
IM	Intersection Matrix
SHARPE	Symbolic Hierarchical Automated Reliability and Performance Evaluator

MCS	Monte Carlo Simulation
FST	Fuzzy set theory

List of Figures

List of Tables:

Thesis Contents

Chapter 1 INTRODUCTION

1.1 Overview

The current technological era is relying heavily on software. It is inevitable to put more emphasis on identifying the risk and implementing the safety features for the real time safety critical embedded systems in a cost effective manner. The most efficient stage to do this is in the early design phase by identifying the risk involved in the system. Identification of risk is the first step of risk assessment and management.

Risk analysis is an important step in the risk management process where consequences of any accident and the probability of the occurrences are identified and analysed in a systematic manner. Probabilistic risk assessment (PRA) is a systematic and comprehensive method to identify and evaluate risks associated with any complex engineered system. According to Mansfield *et al.* (1996a) and HSE(1996), 80% of industrial accidents are caused by risks involved in operation system. Industrial accidents can occur due to human error as well as malfunctioning or failure of equipment, such as: pipeline ruptures, vessel ruptures, chemical releases, deterioration, design faults of a system and software/server failure/malfunction (Pula, 2005). Table-1.1 shows a few examples that illustrate how severe and devastating an industrial accident can be.

Table 1.1: Example of accidents and its results

Place of occurrence	*Date*	*Result*
Flixborough, England *(a vapor cloud explosion)*	June, 1974	28 people dead. The whole plant destroyed and many people injured.
Bhopal, India *(a poisonous vapor burst from a pesticide plant)*	December, 1984	More than 2000 people died and 20,000 injured.
Pasadena, Texas *(a massive explosion)*	October, 1989	23 people died and injured 314 civilians. capital loss of over $715 million
Texas *(a series of explosions in British Petroleum)*	March, 2005	killed 15 people and injured over 170 persons

Every year, major and minor industrial accidents are causing billions of dollars loss to the company and society. There are some safety management organizations and standards in place, such as Occupational Safety and Health Administration's (OSHA), Environmental Protection Agency's (EPA), Process Safety Management (PSM) standard and Risk Management Program (RMP). An automated tool for risk analysis would help industries to do the risk analysis smoothly and reduce the chance of any accident.

1.2 System Safety Terminology

The following important terms are defined by Jia X. (2000):

- **Risk:**

"Risk is the possibility of something undesired occurring. Usually this refers to harm to person, property, or the environment, but can refer to any tangible or intangible loss. Safety is relative freedom from those risks. Risk is measured by considering the likelihood of the undesired events and the magnitude of the attendant losses."

- **Mishap:**

"A mishap is an unintended event or series of events that results in a loss. An informal synonym would be 'accident'. Example of mishaps associated with a nuclear weapon system would include accidental launch of a nuclear missile, damage requiring repair or replacement of the weapon, the unplanned destruction of a nuclear weapon, the radioactive contamination of a nuclear installation and its vicinity, and a nuclear missile boomeranging back to friendly territory. A mishap for an air traffic control system would be a collision between aircraft or between an aircraft and a terrain. For a chemical processing plant, mishaps include intoxication and burns to personnel."

- **Hazard:**

"A hazard is a state of a system or physical situation that, when combined with certain environmental conditions, could lead to a mishap. No accident or loss has necessarily occurred. A hazard is a prerequisite to a mishap. Whenever the hazard is present, the possibility of Mishap exists. Safety is defined in terms of hazard rather than mishaps because mishaps are caused by multiple factors, and only some of those factors may be controlled by the system in question. The existence of hazardous state does not mean that a mishap is inevitable."

It is important to note that during the risk and hazard analysis, the whole system is considered. Any component of the system, such as hardware, different events of the system, interaction between other components, connections, environments, and external conditions are investigated to identify risk or hazard. Probability of mishap and severity of mishap can be

paired together to express risk. Calculating the probability of hazard leads us to the next step where probability of mishap is identified given that the hazardous state exists. Calculating the probability of hazard cannot define risk; it needs to define losses as well.

MILSTD-882(military), NHB5300.4(NASA), DEO5481(Nuclear) have categorized the severity of mishap as follows(Jia X, 2000):

Table 1.2: Category of mishap

Severity of Mishap	***Effects on:***	***Result(s)***
I. Catastrophic	Personnel	Death
	Facilities/equipment/ vehicles	System loss, repair impractical, requires salvage or replacement, severe environmental damage
II. Critical	Personnel	Severe illness or injury requires admission to health care facility lengthy convalescence and permanent impairment.
	Facilities/equipment/ vehicles	Major system damage, loss of primary mission capability, major environmental damage.
III. Marginal	Personnel	Minor injury/illness (medical treatment but no permanent impairment).
	Facilities/equipment/ vehicles	Loss of no primary mission capability, minor environmental loss.
IV. Negligible	Personnel	Superficial injury/illness (little or no first aid treatment).
	Facilities/equipment/ vehicles	Loss time is less than one day. Less than minor system on environment damage.

1.3 Risk Analysis Methodology

Risk analysis is conducted qualitatively and quantitatively. Qualitative analysis involves identifying the possible hazards with the relevant cause; while quantitative analysis involves in investigating the consequences of those hazards in deterministic or probabilistic manner. A variety of risk assessment techniques exist including HAZOP, HAZID, Fault Tree Analysis, Failure Mode Effect Analysis, What If Analysis and Event Tree Analysis, PRA, QRA, LOPA etc. Techniques to identify and rank hazards quantitatively include DOW indices, Mond indices, HIRA index, SweHI, IFAL index.

Some well known risk assessment methodologies are (Ferdous *et al.*, 2009)-

- ***WHO methodology:***
  - i. Identification of hazards
    - (1) Checklist
    - (2) Matrix diagram of integration
  - ii. Assessment of hazards
    - (1) Accident sequence analysis
    - (2) Failure effect analysis
  - iii. Accident consequence analysis
- ***ISGRA methodology :***
  - i. Hazard identification
  - ii. Consequence analysis
  - iii. Quantification of risk
- ***Quantitative risk analysis :***
  - i. Hazard identification

ii. Frequency estimation

iii. Consequence analysis

iv. Measure of risk.

- ***Fault Tree Analysis:***

i. Hazard identification

ii. Top-event identification

iii. Frequency estimation

iv. Measure risk

Fault Tree Analysis is the one of the efficient technique for safety/ risk analysis.

1.4 Brief Description of FTA

In 1961-1962 H. A. Watson along with A. Merans introduced Fault Tree Analysis (FTA) at Bell Telephone Laboratories. It was introduced to study the Minuteman Missile launch control system for US air force. Since then this technique has been extensively explored by design engineers for PRA. FTA is an analysis technique which is basically the visual representation of any possible causes of an accident. This technique assists engineers to measure, identify and evaluate failure, reliability and availability of a complex system. A complex system can be the combination of human and technological entity. There can be different types of nodes in fault tree, such as: **B**asic **E**vent (***BE***), **G**ate **E**vent (***GE***), **C**ondition **E**vent (***CE***), **T**ransfer **E**vent (***TE***). ***BE*** is denoted by the circle-symbol which represents an event that describes a cause of the component failure. A ***GE*** is a logical operator that permits or inhibits fault logic between inputs (lower events) and a single output (higher event). A Fault tree can have five different types of ***GE***s: AND, OR, Exclusive-OR, Priority AND and Inhibit gate. A ***TE*** is denoted by a triangle

symbol and is the indicator of the sub-tree branch (transfer in/out) of the current tree. A ***CE*** is denoted by a oval symbol and use to indicate any specific condition or restriction that may apply to the logic gate.

1.5 Motivation

A typical FTA for an industrial system involves several steps, which require large expert time, precise probability data, and computational capabilities. Many commercial and academic computer-aided FTA tools, including SHARPE, CARA fault tree, PROFAT, Relex Reliability Software, and Fault tree+ have been developed to assist the time-consuming steps of a FTA. The basic steps, i.e., the rules of graphical network construction, the fault tree development and computational time optimization for analyzing a fault tree are more or less same for all FTA tools. However, a significant difference is observed in evaluation of a fault tree when uncertainty due to imprecise and unavailable data becomes a major concern. In traditional FTA, the probability data of basic events are expected to be precise and assigned as crisp or deterministic input, which are often hard to come by for a real industrial system. The crisp failure probabilities of such basic events are difficult to measure and the accuracy of such estimation is often questionable. This is because in practice, especially in the early design stage of a system, there is usually not enough data available. Moreover, many basic events of a fault tree may not have any quantitative or probability data at all. Expert's judgment/ knowledge in this situation is often used as an alternative to attain the objective data (Yuhua and Datao, 2005). However, it comes at the cost of possible uncertainties related to incompleteness (partial ignorance), imprecision (subjectivity), and lack of consensus (if multiple expert judgments are used). Most of the existing FTA tools do not handle this kind of uncertainty and imprecision in input data. In essence, the motivation of this work lies in following three points:

- Lack of an integrated approach to deal with incompleteness in the information for fault tree analysis.
- Lack of an integrated approach to deal with imprecise and subjectivity in the information for fault tree analysis.
- Lack of a computer aided solution with the advanced features as mentioned above to do probabilistic analysis of complex process system.

1.6 Research Objective and Novelty of the Work

In an attempt to circumvent the uncertainty associated in the expert's knowledge, Ferdous *et al.* (2009a, 2009b) have introduced two different approaches (i.e., fuzzy-based and evidence theory-based approaches) to facilitate the accommodation of expert judgment/ knowledge in evaluation of a fault tree, event tree and bow-tie analysis. In this thesis, a software tool – **Expert** knowledge based **F**ault **T**ree **A**nalysis (***ExpertFTA***) is introduced. ***ExpertFTA*** is a sophisticated engineering software tool which incorporates a formal deductive procedure to draw and analyse a fault tree by utilizing Fuzzy set theory and Dempster–Shafer Theory (DST) of Evidence for addressing and handling the uncertainties. In order to be close-to-accurate and build the confidence into FTA, aggregate knowledge from multiple experts is introduced in this tool.

A software-tool should be easily maintainable and reusable. An efficient design and the object oriented approach for a software development can assure reusability and maintainability. Several established design patterns are implemented and object oriented concept of Java, XML and XSLT are used in ***ExpertFTA***.

The novelty of this work can be stated as follows:

- Developed a unique advanced computer aided tool to perform Fault Tree Analysis.
- Implemented Fuzzy set theory along with evidence set theory (i.e. DST) for addressing and handling the uncertainties.
- Introduced knowledge aggregation from multiple experts for individual basic event.
- ***ExpertFTA*** is engineered in such a way that it can be further enhanced with more functionality.

1.7 Thesis Outline

This thesis consists of six chapters. The current chapter has introduced risk analysis methodology and its importance and current practices in the industry. Further, the system safety terminology and the brief description of the FTA technique were discussed. The motivation and research objectives of this work were mentioned in the later section of this chapter.

The second chapter concentrates on the details of FTA steps and the implementation of those steps into software. This chapter discusses the importance of implementing the software in a proper manner by adopting certain design patterns. From the software evolution and maintenance point of view, acquiring several techniques and models are identified and discussed. Later the reason for choosing an appropriate method for current purpose is described. DST and Fuzzy logic is briefly mentioned in this chapter as well. The third chapter illustrates the design and architect of the ***ExpertFTA.*** In chapter four, the development and implementation details of the ***ExpertFTA*** are discussed. Chapter five presents a comprehensive case study and results using this tool. A comparison of ***ExpertFTA*** against a few other available FTA-tools, conclusions from

the current work and a list of future enhancements and directions are provided in the sixth chapter.

Chapter 2 BACKGROUND REVIEW

This chapter provides a review of different background topics related to this thesis. A brief description of FTA (section 2.1) along with DST and Fuzzy set theory (section 2.3 & 2.4) are provided. Latter, software engineering (section 2.5) from a design pattern and model point of view is described. Lastly, a brief overview of XML is provided.

2.1 Overview of Fault Tree Analysis (FTA)

FTA is frequently used in industry to analyze risk and safety. Quantitative and qualitative analysis is the main focus of FTA from accident point of view. Following is an overview of FTA.

The root of the logic tree begins with the undesired effect, which is known as the top event. After identifying the top event, all possible causes for that event is identified. There can be a sequence of events which eventually lead the occurrence of the top event. Those events are drawn by using logic symbols along with the probability of occurrence. According to Haasl, 1965, McCormick, 1981, Roberts *et al.,* 1981, Hauptmanns, 1988, Henley and Kumamoto, 1981, Billington and Allen, 1986, Lees, 1996 , Khan and Abbasi, 1999 and AIChE, 2000, a software tool for FTA should considers following basic steps:

i) Gather information about the system.

ii) Identify the top event.

iii) Gather probabilities of failure of basic events.

iv) Construct the fault tree using logic symbols.

v) Perform quantitative, qualitative and sensitivity analysis.

FTA as a software tool has been available since in early 1970's (Henley, 1981). There are some commercial software tools available, such as: CARA-Fault Tree (CARA-Fault Tree, version 4.1, 1999), Fault Tree+ by Isograph Ltd. and Relex Fault Tree. These tools calculate quantitative analysis by using a conventional probabilistic approach. Imprecision in data has been addressed in PROFAT (PRObabilistic FAult Tree analysis) (Khan and Abbasi, 1999). Fuzzy logic is implemented in PROFAT to handle the imprecision in data, but there are some unresolved issues on basic event data fuzzification and calculation of top event probability. Moreover, there is no GUI for this tool in order to draw and visualize the fault tree. Until today, to our knowledge, there is no FTA-tool which has addressed uncertainty and vagueness of data at the same time by implementing DST and fuzzy set theory side-by-side. The primary goal of this thesis is to come up with an ideal software tool which would implement DST and fuzzy set theory from multiple experts' point of view. The following section discusses the uncertainty in FTA.

2.2 Uncertainty characterization in FTA

Several techniques have been developed to formulate the uncertainty for risk analysis, which are summarized in Table 2.1 (Wilcox and Ayyub, 2003).

Table 2.1: Uncertainty types and formulations

Types	**Nature**	**Techniques**
Aleatory	Stochastic, Objective, Irreducible, Random	Probability theory Evidence theory (random sets)
Epistemic	Imprecise, Incomplete, Ambiguous, Ignorant, Inconsistent, Vague	Fuzzy set theory Evidence theory (random sets) Info-gap theory p-boxes

Uncertainties can be classified into two categories: ***aleatory*** (or stochastic) and ***epistemic*** (or knowledge-based) (Apostolakis, 1990; Thacker *et al.*, 2003; Helton, 2004; Daneshkhah, 2004; Ayyub and Klir, 2006). According to the sources and natures of the uncertainty, ***aleatory*** and ***epistemic*** can be further sub-categorized as *data* uncertainty, *model* uncertainty and *quality* uncertainty (Abrahamsson, 2002; Markowski *et al.*, 2009). Usually incomplete and incomprehensive evaluation of hazards introduces *quality* uncertainty which is also referred to as *completeness* uncertainty (Markowski *et al.*, 2009; Ferdous, 2010). The *data* and *model* uncertainties occur due to insufficient or missing data and consideration of invalid or unrealistic assumptions (e.g., independent); these uncertainties are respectively known as *parameter* and *dependency* uncertainty (Markowski *et al.*, 2009; Ferdous, 2010). Table 2.2 provides detailed descriptions of these categories of uncertainty.

Table 2.2: Source of uncertainty in risk analysis

Steps	Objectives	Techniques	Category of uncertainty		
			Completeness	Modeling	Parameter
Hazard Identification	Identify the possible hazards, develop logic structure of representative accident scenarios (RAS).	HAZOP, PHA, FMEA, Fault Tree and Event Tree	Inability to identify all contributions to risk and all RAS	Wrong interaction between different risk contributors and variables	Imprecision or vagueness in characteristic properties of contributors and variables
Consequence Assessment	Define the possible outcomes, Measure degree of adverse impact on health, property and environmental	Consequence Models	Incorrectness in identification of all types of the consequences as well as of all interactions among consequences	Improper, imprecise and inadequate models for source terms, dispersion and physical effects	Lack or inadequacy or vagueness in values for model variables
Likelihood Assessment	Determine the probability or frequency of RAS	FTA, ETA, bow-tie analysis	Wrong selection of events, safety function and number of accident outcome cases	Wrong analysis and assumptions in FTA, ETA and bow-tie analysis	Limited or unavailable data for components failure rates, events occurrence and interdependent relationships
Risk Characterization	Risk indexes, risk ranking or risk category	Risk matrix, SIL, LOPA	Limited assumptions in external conditions, and incorrect interpretation of results	Inadequacy in selection of appropriate risk measures as well as risk acceptance criteria	Insufficient and limited data on weather conditions, ignition sources and population

In order to address and handle different types of uncertainties conventional or traditional method, probability theory, fuzzy set theory and evidence theory can be introduced. A brief comparison between these three theories is stated below:

The conventional or traditional method is highly desired and well accepted because of its simplicity, input data requirement and minimum analysis time (AIChE, 2000; Abrahamsson, 2002). Though the traditional method is highly desired in FTA analysis, it is incapable of handling any kind of data uncertainty, which most of the time provides an unreliable analysis (Yang and Suzuki, 1995; Abrahamsson, 2002). Probability theory is the most common method to address random uncertainties (Vose, 2008; Ren *et.al.*, 2009). However, this requires sufficient empirical information to derive the PDF (Probability Density Function)s for the inputs (Hammonds *et al.*, 1994; Wilcox and Ayyub, 2003; Abrahamsson, 2002; Chojnacki, 2005, Ferdous, 2009a). Moreover, the classical MCS (Monte Carlo Simulation) framework cannot differentiate random and subjective uncertainties in the uncertainty analysis (Berztiss, 2001; Abrahamsson, 2002). Using fuzzy set theory and evidence theory, uncertainty analysis can be performed with subjectively assigned fuzzy numbers and basic probability assignment (*bpa*)s by the experts (Wilcox and Ayyub, 2003, Ferdous, 2009). Fuzzy numbers are sufficient to address the subjective uncertainty, when the empirical information is sparse or completely unavailable for the uncertain parameters (Chojnacki, 2005, Ren *et al.*, 2009, Ferdous, 2009). Unlike probability and fuzzy set theory, the *bpa* in evidence theory is appropriate to represent uncertainty associated with ignorance and incompleteness of expert knowledge, and able to generalize the overall uncertainty in a belief interval (Bae *et al.*, 2004; Chojnacki, 2005). In some cases, the fuzzy arithmetic and evidence theory-based formulations are still not well-defined,

which often limits their acceptability in risk analysis. Table 2.3 below provides a summary of different types of uncertainty formulations.

Table 2.3: Summary of different types of uncertainty formulations

Characteristics	Traditional	Probability theory	Fuzzy set theory	Evidence theory
Analysis complexity	*H	L	M	M
Data requirement	H	H	M	M
Handling data uncertainty due to subjectivity	L	L	H	M
Handling data uncertainty due to incomplete and inconsistent information	L	L	M	H
propagating different uncertainties	L	M	M	M
Simplicity in the interpretation of results	H	M	L	L
Data aggregation	L	L	M	H
Analysis time	H	L	M	M
Theory acceptance	H	M	L	L

*L: Least desired; M: Moderately desired; H: Highly desired (Ferdous, 2010)

2.3 Evidence Theory Fundamental

Evidence Theory is also known as DST as it is developed by Dempster(1967) and later extended by Shafer (1976). The major motivation worked behind the development of this theory was to characterize the uncertainty caused by partial ignorance, knowledge deficiency or inconsistency (Sadiq *et al.*, 2008; Wang *et al.*, 2004). Unlike traditional probability theory and Fuzzy set theory, evidence theory distributes the subjective knowledge of an expert to the corresponding subsets of a power set. The unassigned mass due to partial ignorance or

incomplete information is assigned to an ignorance subset within the power set (Ferdous, *et al.*, 2011; Sadiq *et al.*, 2008).

The major advantages of using the evidence theory include (Sentz and Ferson, 2002):

i) Individual beliefs from different sources can be expressed through the probability mass function that may bear incompleteness from partial to full ignorance,

ii) A belief interval (a boundary of probability estimation) can be obtained for each uncertain parameter, and

iii) Bias from a specific source can be avoided and conflicts among different sources can be resolved through a belief structure.

Evidence theory generalizes classical probability theory through a belief interval constructed by assigning upper and lower bounds for probabilities (Guth, 1991). It uses four basic constituents: *frame of discernment (FOD)*, *basic probability assignment* (*bpa*), *belief* measure (*Bel*), and *plausibility* measure (*Pl*) to characterize the quality of uncertainty, such as probability of basic-events, events or input events (Sadiq *et al.*, 2008). The theory also includes reasoning based on the rule of combination of degrees of belief according to different evidence.

For a given FOD, *(Ω)* in Figure 2.1, *bpa (mass)* is distributed over the set of all possible subsets of Ω: the power set of Ω and written 2^{Ω}. The unassigned mass, calculated by 1- *m(p)* -*m* *($\neg p$)*, is assigned to the belief mass for the ignorance subset. The basic formulations for different parameters of evidence theory are stated in the following sub-sections.

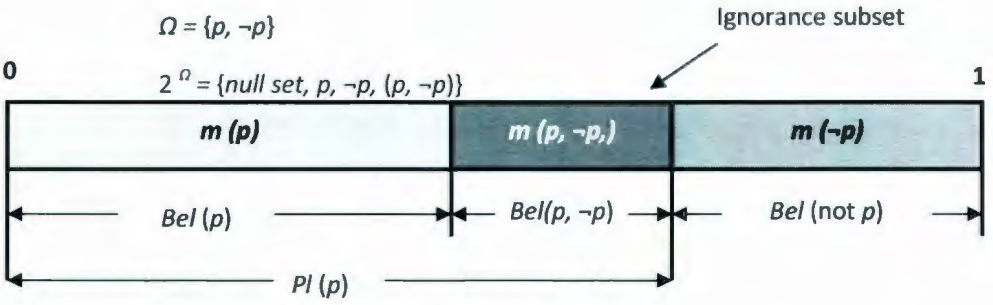


Figure 2.1: Formulation of uncertainty using evidence theory

2.3.1 Basic Formulations of DST

The total subset of a power set (P) in the DST theory is determined by 2^{Ω}. For example, if the number elements of a FOD is two i.e., (Ω) = {T, F}, then the power set (P) comprises of four subsets, i.e., {(Φ, a null set), (T), (F) and (T, F)}.

The *basic probability assignment* (*bpa*) in DST theory represents the knowledge proportion of every subset (p_i) in the power set (P) such that the sum of the proportion is 1. The *bpa* is denoted by $m(p_i)$ and can be defined with the Equation 1:

$$m(p_i) \rightarrow [0,1] \; ; m(\Phi) = 0 \; ; \sum_{p_i \subseteq P} m(p_i) = 1 \qquad (1)$$

The lower probability bound- *belief(Bel)*, for a set p_i, is defined as the sum of all the *bpa*s of the proper subsets p_k that support the minimum knowledge of interest p_i, where $p_k \subseteq p_i$. The *Bel* is written as:

$$Bel(p_i) = \sum_{p_k \subseteq p_i} m(p_k) \qquad (2)$$

The upper probability bound, i.e., the *plausibility* (*Pl*) measure for a set p_i is the summation of *bpa*s that support the maximum knowledge including the ignorance subset. Therefore, the relation can be written as:

$$Pl(p_i) = \sum_{p_k \cap p_i \neq \Phi} m(p_k) \qquad (3)$$

2.3.2 Knowledge Combination Rules for DST

The combination rules in DST allow aggregation of different degrees of belief from different expert's knowledge and provide a combined belief structure (Ferdous *et al.*, 2010). The Dempster and Shafer (DS) rule is fundamental among all combination rules in evidence theory. A number of modifications of the DS rule on the basis of minimization and normalization of conflicts among the different sources have been reported (Sentz and Ferson, 2002; Sadiq *et. al.*, 2008). The most common modifications include those by Yager, Smets, Inagaki, Dubois and Prade, Zhang, Murphy, and more recently by Dezert and Smarandache (Sadiq *et al.*, 2008). Detailed discussion and comparisons of these rules can be found in Dezert and Smarandache (2004). To address two extreme cases of conflictions, high-conflict and non-conflict issues in the experts' knowledge, DS and Yager combination rules are used in this study. The details of these two rules are given below.

i) DS rule of combination: Dempster's *rule of combination (DS)* uses a normalizing factor (1-*k*) where '*k*' is the sum of all *bpa*s with conflict. This method ignores the conflicts between two sources (eg. m_1, m_2) by dividing the combined evidence with

that factor. DS combination rule uses Equation 4 to combine evidences from different sources:

$$[m_1 \oplus m_2](p_i) = \begin{cases} 0 & for\ p_i = \Phi \\ \dfrac{\sum_{p_a \cap p_b = p_i} m_1(p_a) \times m_2(p_b)}{1-k} & for\ p_i \neq \Phi \end{cases} \quad (4)$$

ii) Yager rule of combination: Yager(1987) proposed a similar method as DS but more robust where the degree of conflict is very high among the sources (Sentz and Ferson, 2002). The difference between DS and Yager rule is that the degree of conflict (k) in Yager is not used for normalization. It is directly added up with part of ignorance **Ω**. The equations uses in Yager are as follows:

$$[m_1 \oplus m_2](p_i) = \begin{cases} 0 & for\ p_i = \Phi \\ \sum_{p_a \cap p_b = p_i} m_1(p_a) \times m_2(p_b) & for\ p_i \neq \Omega \\ \sum_{p_a \cap p_b = p_i} m_1(p_a) \times m_2(p_b) + k & for\ p_i = \Omega \end{cases} \quad (5)$$

2.3.3 "Bet" Estimation

The interval obtained from the *belief* and *plausibility* measures gives the belief structure of expert knowledge. The belief structure takes into account the ignorance and conflicts in multi-expert knowledge and provides a range for the event probability. *"Bet"* estimate in DST

provides a point estimate in belief structure (similar to defuzzification), which is estimated by Equation 6. In equation 6, $|p_i|$ refers the cardinality (number of elements) in the set p_i.

$$bet(P) = \sum_{P \subseteq p_i} \frac{m(p_i)}{|p_i|} \tag{6}$$

2.4 Fuzzy Set Theory (FST) Fundamental

In 1965, Lotif Zadeh introduced his pioneer work: fuzzy set theory, where he argued that the conventional probability theory is not sufficient to express the intensity (degrees) of truth in subjective information. It introduced robustness into QRA (Quantitative Risk Analysis) by allowing a certain amount of imprecision to exist, thus paving the way to represent human linguistic terms as fuzzy sets, hedges, predicates and quantifiers (Rivera *et al.,* 1999). Many disciplines including control systems, neural networks and artificial intelligence have adopted this concept. Fuzzy set theory (FST) is flexible enough to translate the expert's linguistic variable in probability domain and deal with subjective uncertainty due to imprecision or vagueness in data.

FST uses the fuzzy number to describe the relationship between an uncertain quantity p (e.g., event probability) and degree of uncertainly through membership function μ. Any type of membership function including normal, bounded and convex functions, e.g., triangular, trapezoidal and Gaussian shapes, can be considered for the formation of a fuzzy number. However, the selection of a function essentially depends on the variable characterization and available information (Ferdous, *et al.*, 2010). The TFN (Triangular fuzzy number) can be

described by a vector (p_L, p_m, p_U) that represents the lower boundary, most likely value, and upper boundary of the uncertain quantity. The α-cut for a TFN represents the degree of membership of p_f in the set P. The membership function of a TFN can be described as Equation 7:

$$\mu_P(p_f) = \begin{cases} \dfrac{p_f - p_L}{p_m - p_L} & p_L \le p_f \le p_m \\ \dfrac{p_U - p_f}{p_U - p_m} & p_m \le p_f \le p_U \\ 0 & otherwise \end{cases} \tag{7}$$

2.4.1 Fuzzy Arithmetic Operations

The α-cut based formulations are simplified and commonly used arithmetic operations of FST (Lai *et al.*, 1983; Siler *et al.*, 2005; Li, 2007). These operations are performed at each corresponding membership value of a α-cut for adding, subtracting, multiplying, and dividing fuzzy numbers to determine overall operation results on the fuzzy sets (Wilcox and Ayyub, 2003). Fedous *et al.* (2009a) explored fuzzy arithmetic operations for describing the dependent and independent relationship between basic events or events in FTA and ETA. The current ***ExpertFTA*** tool considered that the interdependence among the basic events in FTA is independent. In other words, basic events are independent in ***ExpertFTA***. In Table 2.4, the intersection and conjunction rules for "AND" and "OR" operations for FTA are shown.

Table 2.4: α-cut based fuzzy arithmetic operations for FTA

Rule	Operation	α-cut formulation
Conduction	"OR" gate	$p_L^{\alpha} = 1 - \prod_{i=1}^{n} (1 - p_{iL}^{\alpha})$; $p_R^{\alpha} = 1 - \prod_{i=1}^{n} (1 - p_{iR}^{\alpha})$
Intersection	"AND" gate	$p_L^{\alpha} = \prod_{i=1}^{n} p_{iL}^{\alpha}$; $p_R^{\alpha} = \prod_{i=1}^{n} p_{iR}^{\alpha}$

2.4.2 Multiple Expert's Knowledge Aggregation

Knowledge aggregation from multiple experts becomes essential in an analysis. It pools multiple sources of knowledge into one and provides a mutual agreement of different sources of knowledge (Lin and Wang, 1997). A number of methods, e.g., max-min, arithmetic averaging, quasi-arithmetic means, weighted average method, fuzzy Delphi method, symmetric sum and t-norm, are available to aggregate multiple experts knowledge in the form of fuzzy numbers (Huang *et al.*, 2001; Sadiq *et al.*, 2007; Wagholiar, 2007). The weighted average method is the simplest method allowing aggregation according to prior weights of the arguments. It uses the following equation for aggregating *m* experts knowledge.

$$P_i = \frac{\sum_{j=1}^{m} w_j P_{i,j}}{\sum_{j=1}^{m} w_j} \qquad i = 1,2,3,.....n \tag{8}$$

where P_{ij} is the linguistic expression of uncertain input basic event i elicited from expert j, n is the number of input events, m is the number of experts, w_j is a weighting factor corresponding to expert j and P_i is the aggregated fuzzy number. For equally weighted

knowledge, the weighted average method gives a similar estimation to the arithmetic averaging method.

2.4.3 Defuzzification

Defuzzification in FST transforms a fuzzy number into a crisp value (Klir *et al.*, 2001). Many defuzzification methods are available in the literature (e.g., Klir *et al.*, 2001; Ross, 2004). The weighted average method is a computationally efficient method (Ross, 2004; Khan *et al.*, 2005). The following equation is used for defuzzification of outcome event probability or frequency.

$$P_{out} = \frac{\sum \left[\mu_P(\tilde{P}) \cdot \tilde{P}\right]}{\sum \mu_P(\tilde{P})} \tag{9}$$

2.4.4 Alpha-cut Determination

The α-cuts are used to determine fuzzy intervals (i.e., nested intervals in a fuzzy number) with a membership grade (μ_P) greater or equal to the α-cut value (Wilcox *et al.*, 2003). In a TFN, the membership function uses the following relationship to determine the interval at the α-cut level:

$$\tilde{P}_\alpha = [p_L + \alpha(p_m - p_L), p_R - \alpha(p_R - p_m)] \tag{10}$$

2.4.5 Fuzzy Linguistic Variable

Fuzzy linguistic variables are linguistic terms to articulate uncertainty in probability estimation. Experts' knowledge can be expressed in terms of linguistic variables such as very high, high, medium, very low, low, etc. (Ayyub,1991; Wu,2006, Sadiq *et al.*, 2007).

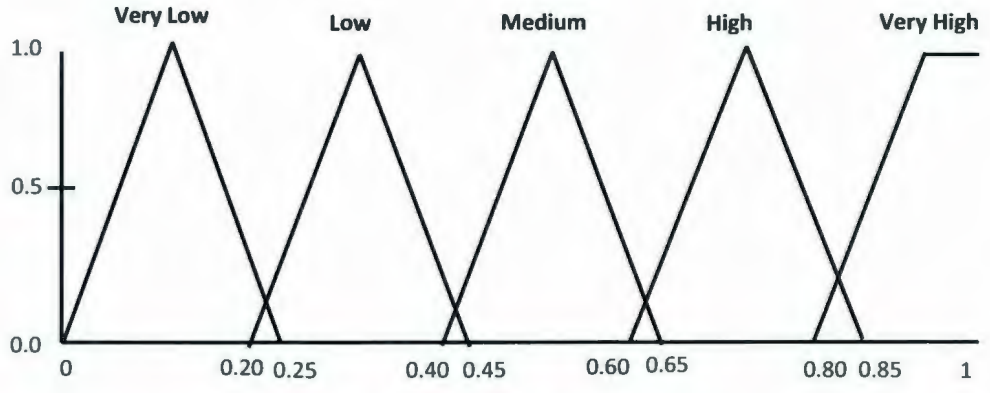


Figure 2.2: Example of fuzzy linguistic scale

2.5 Discussion on Software Development

In order to implement the theories described above, along with a GUI (Graphical User Interface), it is important to deal with software development carefully. It is imperative to adopt a solid software engineering technique in the very early stage of software development. A concrete software engineering technique involves considering or choosing appropriate design pattern(s) and process model. The following sub-sections briefly discuss design patterns and process models and also, Object Oriented Programming (OOP) concepts.

2.5.1 Design Patterns

Christopher Alexander articulated the concept of *patterns* which he proposed in his book *The Timeless way of Building* (Alexander, 1979) and *A Pattern Language- Town, Buildings, Constructions* (Alexander *et al.* 1977). The essence of his concepts is that every recurring problem follows some patterns. Certain numbers of patterns are needed in order to confine the essence of all architectural designs. Unlimited architectural design can be accomplished by adopting and combining those patterns. Software design also resembles architectural design. The solution of the most commonly encountered problems in software design is described in *software design patterns*. The main purpose of using software design patterns are as follows:

i) To help software designers by providing summarized experience from various software designs in a few design patterns.

ii) To increase the confidence of the designer in order to reuse the proven design in software systems.

iii) To provide common vocabulary for software designers.

Gamma (1995) in *Design Patterns* first introduced twenty three most commonly used software design patterns. Those patterns are divided into three categories: *Creational Patterns, Structural Patterns, Behavioural Patterns.* The third chapter provides the discussion about these categories.

2.5.2 Software Process Model

An abstract descriptive representation for a particular large and complex software system can be defined as software process model (Sommerville, I. 2001). Basically, it is a road map

which includes the descriptions of all necessary steps and tools in order to implement a software product successfully (Bell, D. 2001). To deal with ever increasing size and complexity of software systems, software development process has been improved drastically over time. Software development is no longer just writing code and fixing bugs when it is encountered; rather, with the passage of time, it is now evolved to a full-fledged comprehensive life cycle. Different phases can be involved in a software paradigm which may include requirement analysis, design, implementation, debugging, testing, deployment and maintenance. In order to handle different circumstances, different process models can have more or less phases and in different orders. That is why, based on the project's nature and constraints, an appropriate model should be selected. There are four major and most common process models which are briefly described below:

i) Waterfall model:

In 1970, this model has introduced by Royce (Royce, W.W. 1970). This model consists of requirement gathering, design, implementation, testing, deployment and maintenance phase. This is a sequential process where in order to proceed to the next phase; earlier phases have to be finished. Hence, the main disadvantage of this model is that, if any error occurs, it can only be identified during deployment phase. Only the fully completed version is given to the user. Lack of end user interaction makes it hard to get any comments from the user in earlier stages.

ii) Incremental model:

Basili, V.R. and Truner, A.J. introduced the concept of the incremental model in 1975. This model is essentially the composed of a series of waterfall-model and combines certain phases as a cycle. Here, user requirements are well defined in the beginning of the project. This model is useful for the projects in which user interaction and feedback is important during the development because this model allows users to get an evaluation version of the project in almost every cycle.

iii) Prototyping model:

This model was introduced by Floyd, C. in 1984 which also known as the evolutionary model. This model is based on the concept of improving the prototype system in multiple iterations. Specification, development and testing are done here concurrently and the final version of the software fulfills the requirements of the user.

iv) Spiral model:

For high risk and complex projects this model is suggested, although it is a bit complex and costly. This model was introduced by Boehm, B.W in 1988 where he combined the features from the above three models. By combining those features, this model enhances the users' interaction and risk management at the same time. For the project in which requirements might change frequently and it is crucial to be verified by the end-user, this model is ideal.

For ***ExpertFTA*** purpose, the requirements are well defined, as well as the theories which will be implemented. Users' interaction is hardly needed in the implementation phase. The Waterfall model is chosen for this software. In addition, ***ExpertFTA*** is designed in such a way so that future enhancement will be very easy.

2.5.3 Object Oriented Programming (OOP)

Programming and design are two distinct tasks in the object-oriented paradigm; however, they are more tightly interlinked than in the conventional programming paradigm. Capture system's desired behaviour by using notations (such as the Unified Modeling Language: UML) is the main goal of *object-oriented analysis and modeling;* on the other hand, *object-oriented design* concentrates on creating an architecture for implementation. One of the fundamental principal of the object-oriented approach is *modularization;* a software system is decomposed into highly cohesive and loosely coupled modules.

In OOP a *class* (module that includes data structure, functionalities/behaviours and state of the module) can be instantiated multiple times as required and each instance is referred to as an *object*. A particular *object* can also be derived or inherited from another *class*. In that way the code redundancy can be optimized. Updating or modifying the state of an instance of a particular *object* will affect all of the instances of that *object*. A *method* of an *object* is essentially a placeholder (block of code) to perform a specific task or operation which can be invoked through an instance of that class. Sending and receiving messages to and from an *object* is performed via a method. The advantages of *OOP* are as follows:

- System's behaviour or functionalities can be abstracted and encapsulated through interfaces (contains only method's signature) and classes (which may implements interface(s) or extends another class) with methods respectively;
- An entity or interface can have multiple interchangeable implementations, which is defined as *polymorphism*; relationships among classes and interfaces can be defined through *inheritance*, which eliminates redundant data specification.

- *OOP* provides a solid framework for the software and also improves maintenance, facilitate enhancements and code reutilization.

The concept of *OOP* provides a powerful mechanism for information storage, retrieval and evaluation. Fault-tree analysis is basically a concise tree representation. It consists of different kinds of gates and basic events which can adopt this concept precisely. Due to the fact that a software tool like ***ExpertFTA*** is of moderate complexity contains large number of classes to distribute its logic, computation and I/O functionalities, maintenance and refactoring becomes complicated if it is not designed smartly in the initial phase.

2.6 XML

XML (Extensible Markup Language) is an interoperable, easily readable (both by human and machine) and self-descriptive document. The specification to encode the XML document is defined in XML 1.0 by W3C (The World Wide Web Consortium to develop the web standards). Compared to any relational database, XML is very light-weight, reading and writing is faster, and portable to different platforms without any difficulties. In addition to all those advantages, a particular database does not need to be installed and maintained to get those facilities. A database is mostly useful where large volume of information and different type of relational data needs to be stored in a secured environment. As such, in this case, ***ExpertFTA*** needs to store recent and retrieve previous straight-forward and light-weight information. Hence, XML seems to be the most suitable choice for the current purpose.

Chapter 3 *ExpertFTA* MODELLING AND DESIGN

This chapter describes the design of the ***ExpertFTA*** tool. After giving an overview of fundamental structure and operational steps of computer-aided Fault Tree Analysis, this chapter illustrates the requirements analysis, overview of the GUI and other major phases of development. A brief summary of lessons learned while designing this tool is also given towards the end of the chapter.

3.1 Computer-aided Fault Tree Analysis

From an earlier discussion (in previous chapters), it was concluded that the FTA is a complex technique which essentially requires a user friendly, optimized software tool. In order to develop a suitable tool requires a detailed understanding of software engineering. The FTA-tools developed so far maintain common fundamental structures and operational steps. Figure 3.1 shows basic operational steps adopted in a traditional computer-aided FTA.

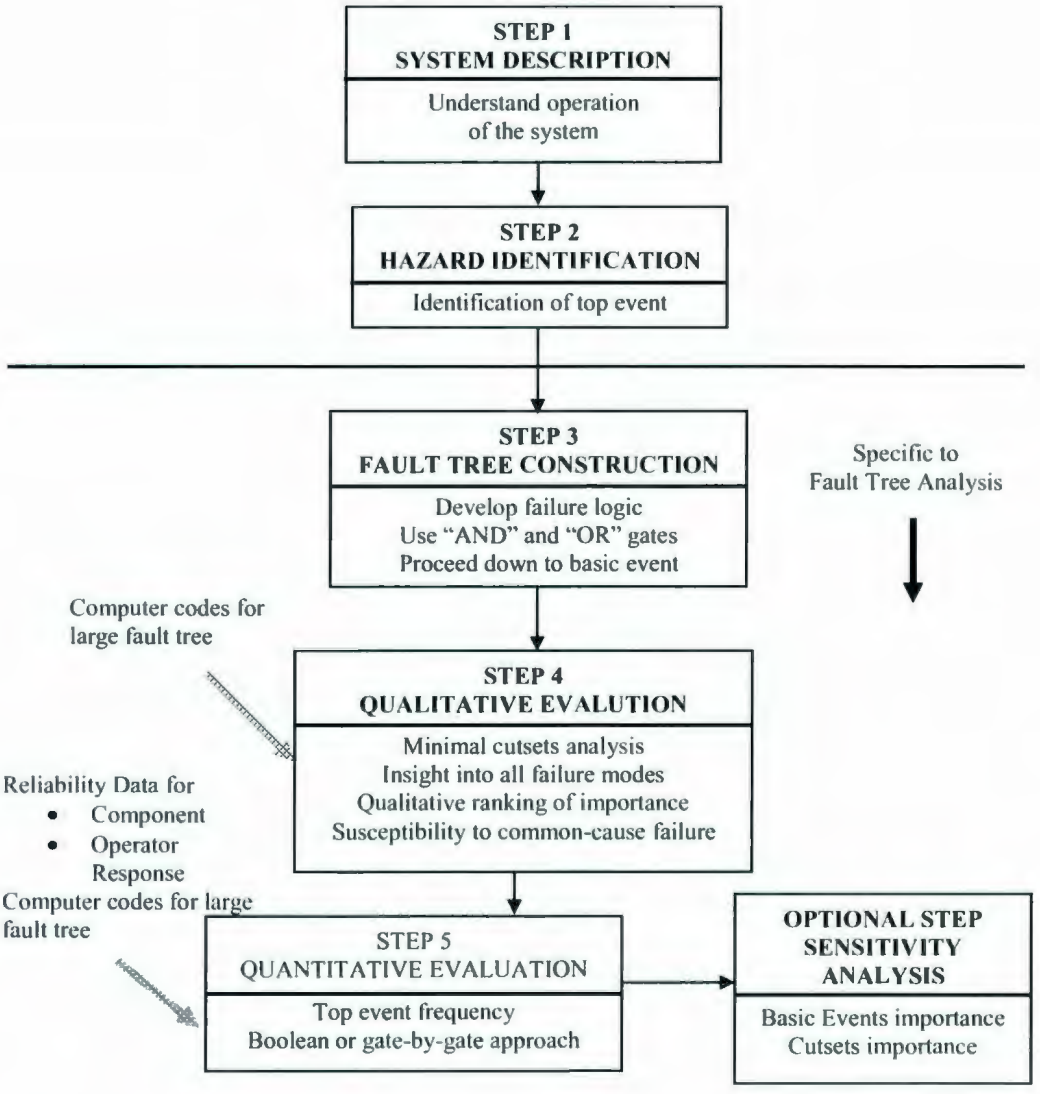


Figure 3.1: Operational steps for FTA (after AIChE, 2000)

Figure 3.2 depicts the fault tree development structure as per Henley *et al.* (1981). This figure illustrates the different type of events used in FTA. The undesired incident (e.g. leakage, fire and explosion) that needs to be further analyzed is treated as a top event and placed on the top of the fault tree.

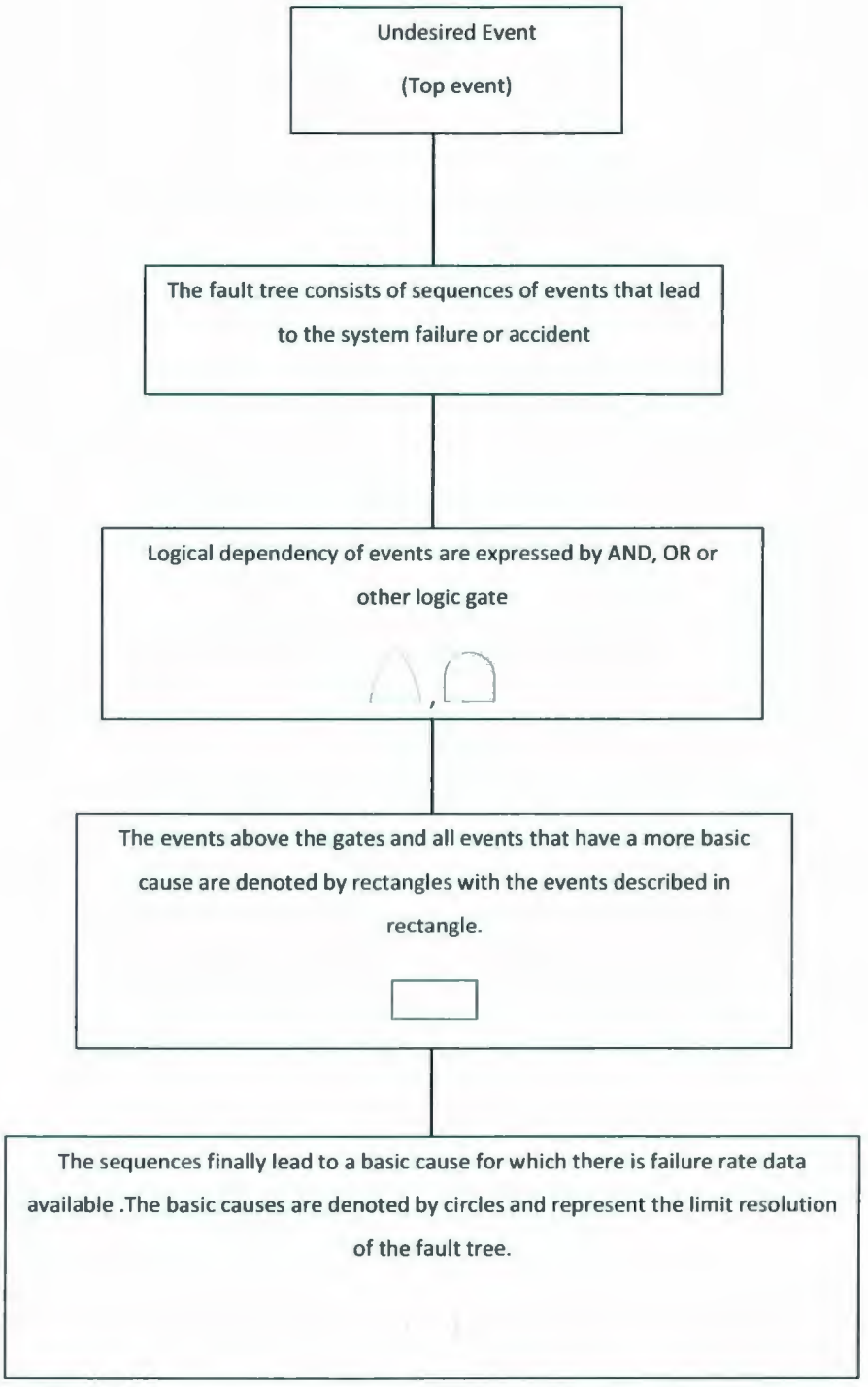


Figure 3.2: Fundamental structural diagram of fault tree (Henley *et al.*, 1981)

Failure or error in a system or component that causes the undesired event is termed as the basic event and placed on the bottom of the tree. Basic events are connected through logic gates. There are different types of logic gates: AND, OR, Exclusive-OR, Priority-AND and Inhibit Gate. A fault tree can also consists of intermediate events, external (initiating) events, undeveloped events, conditioning events and transfer in and transfer out. In the subsequent section, fault tree analysis tool development is discussed from a software engineering point of view.

3.2 *ExpertFTA* Requirements

The primary objective of this work is to implement Fuzzy set theory and DST in FTA. The goal is not to implement all of the features of the fault tree once. It is rather to keep the fault tree as simple as possible and to observe how accuracy can be regained from implementing advanced concepts. To achieve this goal, it is decided to implement the most basic features of a fault tree and keep GUI as simple as possible. That being said, the current version of ***ExpertFTA*** will only include the symbols for basic event, AND and OR gate; implementation of the remaining symbols are not essential to accomplish the present research objective. The intermediate event is not considered, instead, any gate or basic event will be joined with its parent-event directly. In addition to those above points, the following requirements are expected to be handled in ***ExpertFTA***:

1) The fault tree must be analyzed using Fuzzy set theory and DST so the user can compare the results and make a fruitful decision.

2) Following the Fuzzy route, there has to be a fuzzy-scale available to input data. Users need to be able to construct their own fuzzy-scale; so that they won't be restricted by any pre-defined fuzzy-scale.
3) Three different options for input will be available: "Fuzzy Scale", "Most-Likely Value" and "Crisp Value".
4) Ability to aggregate knowledge from multiple experts for both Fuzzy and DST route.
5) In DST, users should be able to choose from different combination methods (e.g, Yegar, DS) for each basic event input.
6) Ability to perform sensitivity analysis in Fuzzy route by alpha-cut evaluation.
7) Result from both routes should be provided as gate-by-gate probability calculation.
8) In DST route, Bet-estimation should also be included with gate-by-gate probability calculation.
9) Any kind of input should be captured and stored into XML file for future reference.
10) Finally, a well-formatted report should be generated by the application.

The following two sections describe the modelling (section 3.3) and design (section 3.4) of ***ExpertFTA*** respectively to accomplish the above requirements.

3.3 *ExpertFTA* Modelling

This section talks about the software modelling of ***ExpertFTA*** from the users interface point of view. To provide the user a simplest form of user-interface, the tabbed-navigation pattern is being selected. Once the user finishes drawing the fault tree and proceed to insert data, one tab will be created (based on the selected logic mode: Fuzzy/DST). Upon finishing assigning the data, the user will be able to view the calculated result in another tab. A similar procedure is followed for both logic modes. If the user uses both logic modes for analyzing the fault tree, he can go back and forth to view any tab at any time. Below a few screen models are described including the wireframe drawing. Wireframe is the graphical representation of the functional overview, often called the blueprint.

3.3.1 "Draw Fault Tree" Interface

The very first tab is dedicated to drawing the fault tree. There are horizontal and vertical palettes which contain different command-buttons and symbols. A canvas is placed at the center of the screen where the fault tree can be drawn. There are a few rules which the user needs to maintain while drawing the fault tree. Otherwise, the application will inform the user to fix it and will not process the request. The rules are:

1) A particular gate has to be selected prior to adding any event underneath it.
2) Only a particular (either AND or OR) gate can be placed underneath the top-event. No more than one gate can be added and also no basic event is allowed for this position.
3) A gate or a basic event cannot be placed underneath any basic event.

4) A gate must contain at least two child-events (might be combination of gate and basic event) associated with it.

ExpertFTA is structured to follow the top to bottom approach. User will start drawing the fault tree from the top event and finish with the basic events.

Below, Figure 3.3 depicts the wireframe of the "Draw Fault Tree" Screen.

Figure 3.3: Wireframe of "Draw Fault Tree" interface

On the bottom of the screen, there is an option-panel from where the user can choose the logic mode ("Fuzzy" or "DST") and press "Finish" button to move to the next screen (i.e. "Input parameter").

3.3.2 "Input Parameter" Interface

The "Input parameter" screen for Fuzzy set theory will be slightly different from DST in terms of appearance and functionality. This screen is divided into two sub-screens. The left sub-screen contains all the basic events information which are present in the fault tree. There is a button (named "Expert's Knowledge...") beside each basic event ID.

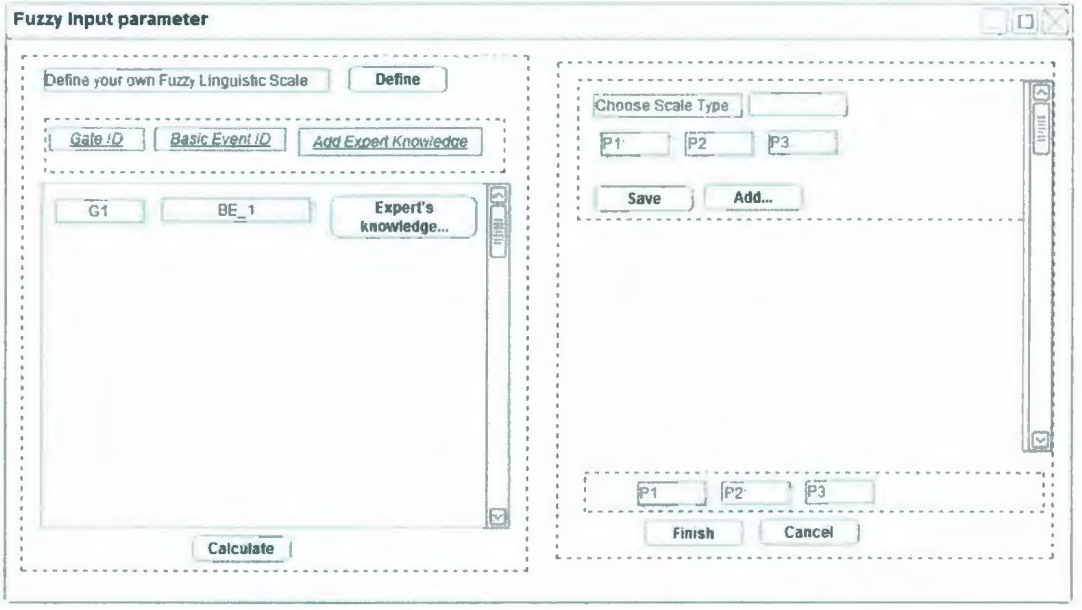


Figure 3.4: Wireframe of Fuzzy "Input Parameter" interface

This button facilitates the user to enter input-value from multiple experts for that particular basic event. In the right sub-screen, once the user adds input-value from a certain expert and clicks the "Save" button, the value is temporarily saved and displayed on the bottom of this sub-screen. By clicking the "Add..." button, the user gets another similar panel underneath the current panel to input values from another expert. Once the user is finished entering values and clicks the

"Finish" button, this will store the calculated value for the current basic event is stored. On the bottom of the right sub-screen there is a "Calculate" button, which calculates the result based on the provided input values for all basic events; if any of the basic events do not have values associated with them, an error message is shown.

In the Fuzzy "Input Parameter" screen, at the top of the right sub-screen, there is a button called "Define" which enables the user to construct a Fuzzy Linguistic scale. Details about the "Fuzzy Linguistic Scale" Screen are discussed in the next section. The user can choose different scale (from Scale-type combo-box) while entering input for each expert. Figure 3.4 above shows the wireframe of the Fuzzy "Input Parameter" screen.

In the DST "Input Parameter" screen, beside the common functionalities, the user can choose frame of discernment as well as combination method (DS or Yeger) to calculate Belief and Plausibility.

3.3.3 "Fuzzy Linguistic Scale" Interface

This screen will facilitate the user to construct a Fuzzy Linguistic scale which can be used during the current session of this application. At first, the user deceides how many categories need to have in this scale and assign a meaningful name for these categories. Afterwords, the user needs to select the type of value for each category (i.e. TFN or ZFN). Upon choosing the value type for categories, the user sees the subsection to assign the boundary values for the categories. Once the user clicks the "Save" button, the application stores it for further use during the current session. Figure 3.5 provides the wireframe of this screen.

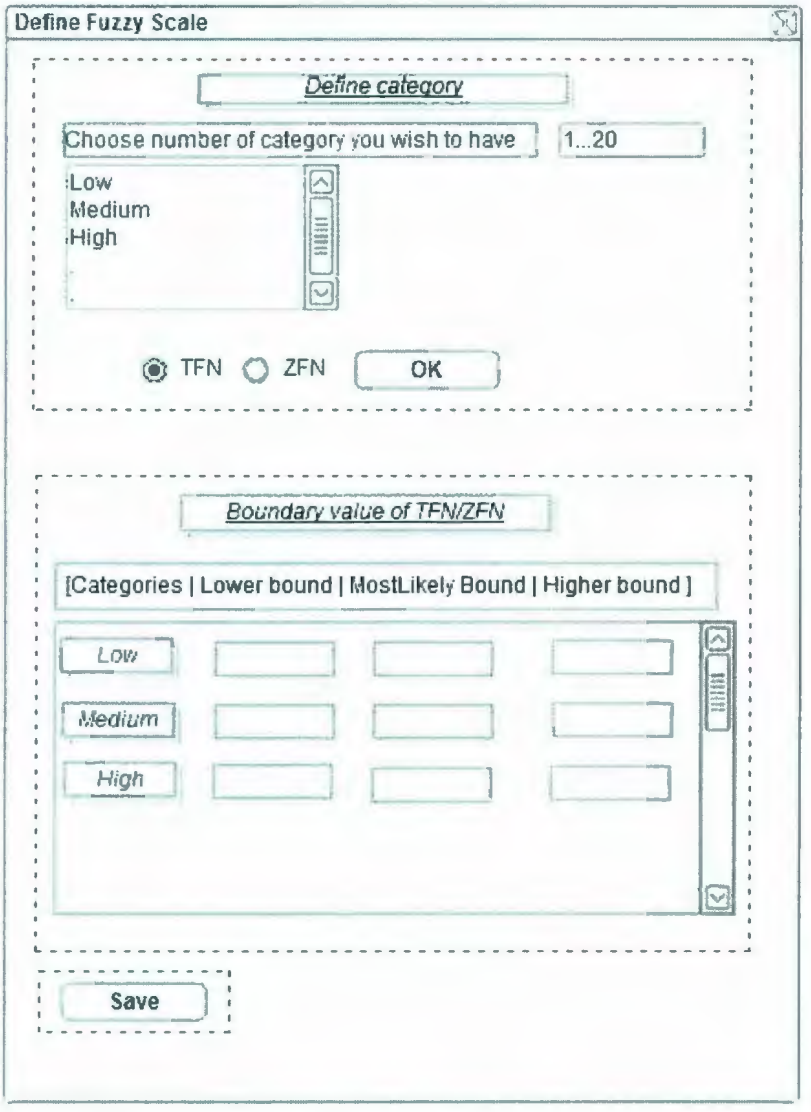


Figure 3.5: Wireframe of "Fuzzy Linguistic Scale" interface

3.3.4 "Analysis" Interface

Once the probability calculation is finished for the entire fault tree, the user can view the result and perform some analysis (depends on the calculation mode that has been selected earlier) on it. For both logic modes (DST and Fuzzy), the left sub-panel contains the results of each gate. In addition to that in DST mode, there will be "Bet" estimation of each Gate as well. On the other hand, in Fuzzy Analysis, Alpha-cut estimation can be performed on the right sub-panel. Finally, the user is able to generate a detailed report on the current fault tree by clicking the "Generate Report" button at the bottom. Figure 3.6 depicts the "Fuzzy analysis" screen.

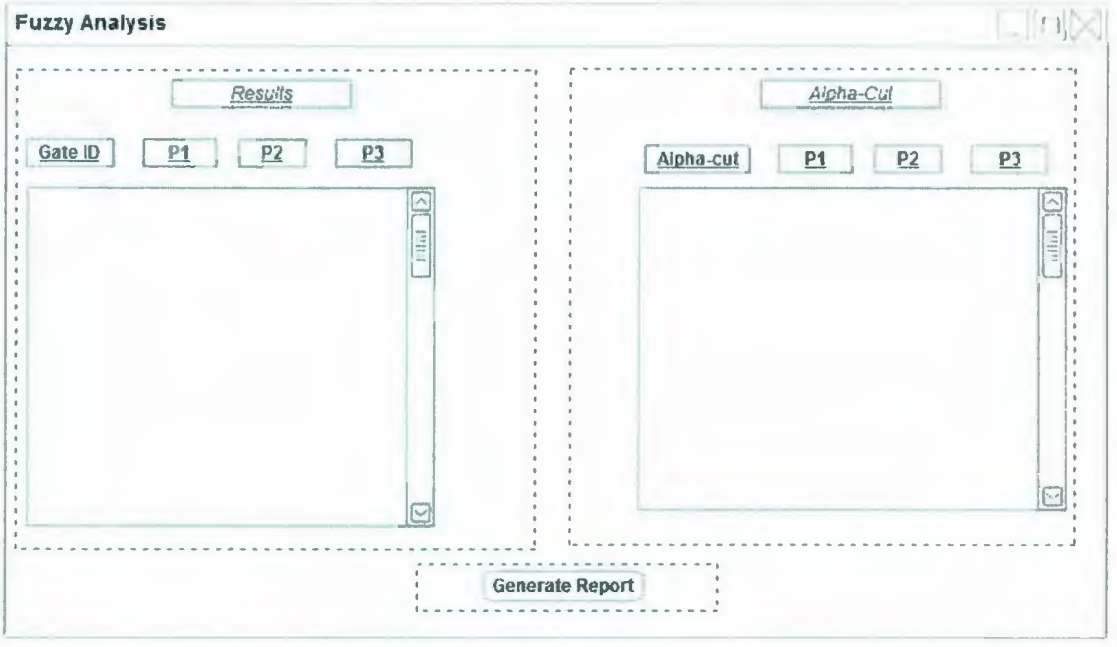


Figure 3.6: Wireframe of "Analysis" interface

3.4 *ExpertFTA* Design

This section is divided into two sub sections to describe *ExpertFTA* design from a formal work-flow and design-pattern point of view.

3.4.1 Work flow of *ExpertFTA*

The work-flow diagram of a particular phase of the application provides a visual overview of different actions or steps which take place in that phase. In this section, different phases of the application are investigated from a design point of view and provides work flow diagram to illustrate these.

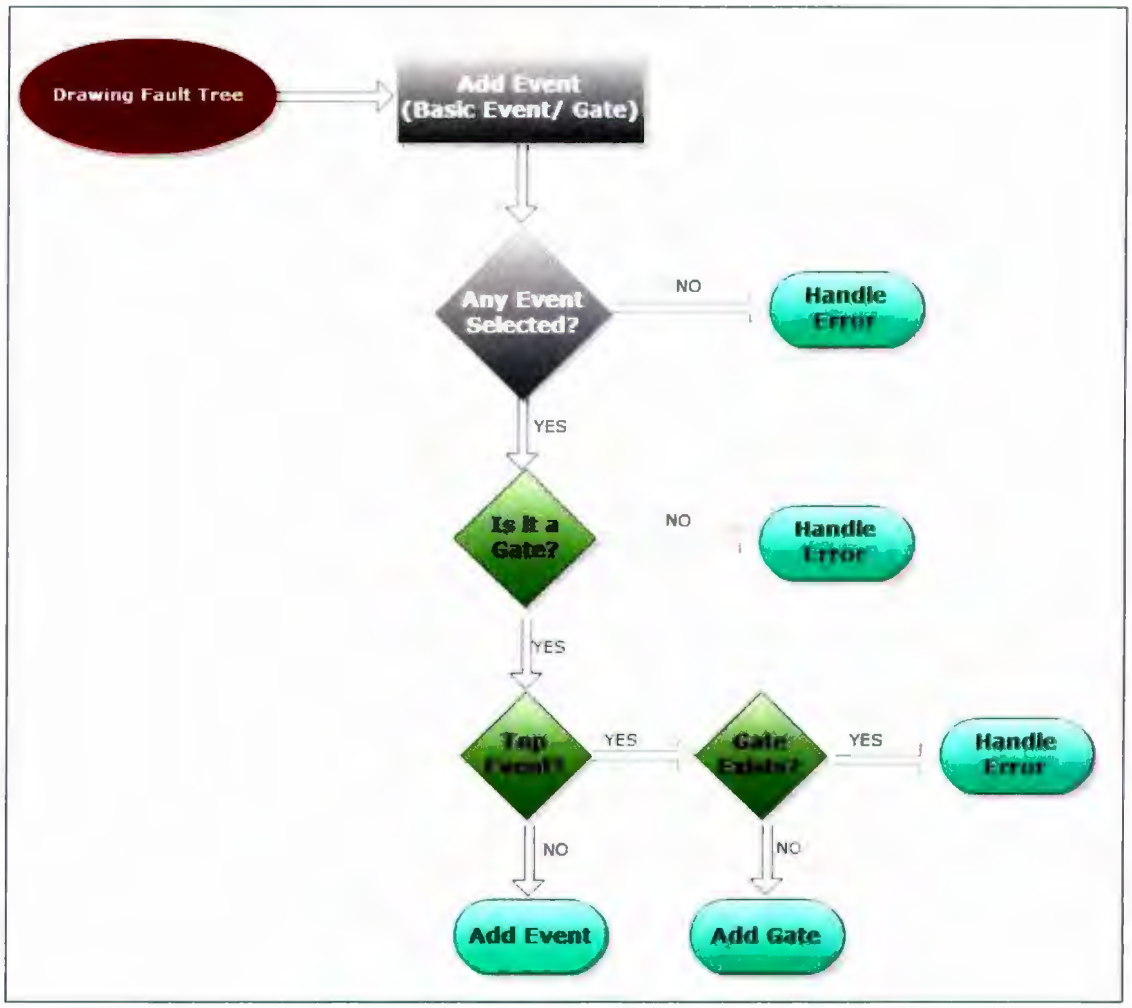


Figure 3.7: Work-flow diagram of drawing tree

Figure 3.7 shows work flow diagram of drawing a fault tree on the canvas. Drawing a fault tree starts with the single gate, not with the basic event. Any gate event must contain at least two or more events (it could be gate or basic event). A basic event cannot have an event as a child associated with it.

Figure 3.8 is a complete overview of the work-flow in ***ExpertFTA***. Once the fault tree is drawn and logic mode is selected, it validates the entire fault-tree before proceeding further. This

is the initial validation phase where it checks whether all gates and basic events are placed appropriately according to the rules.

In Fuzzy mode, the user can define the linguistic scale prior to entering an input value for any basic event. The user can also undo the existing linguistic scale and redefine, if necessary. In order to input a value for any basic event, a particular scale has to be selected. The entered value will be validated prior to saving; knowledge aggregation can be performed by adding additional experts for input values for the same basic event. The user must input value(s) for all basic events in order to calculate the result.

Similar to the Fuzzy mode, in DST, multiple experts knowledge can be inroduced for a single basic event. For each basic event, a frame of discernment (FOD) needs to be selected and for each expert, a combination method (DS/Yager) has to be selected. The application will validate the input values and also check whether all required options are selected prior to saving.

A report will be generated by parsing the XML file and using predefined XSLT (Extensible Stylesheet Language Transformations) dedicated to the report. The application will validate the XML file prior to parsing it and impose the XSLT on it.

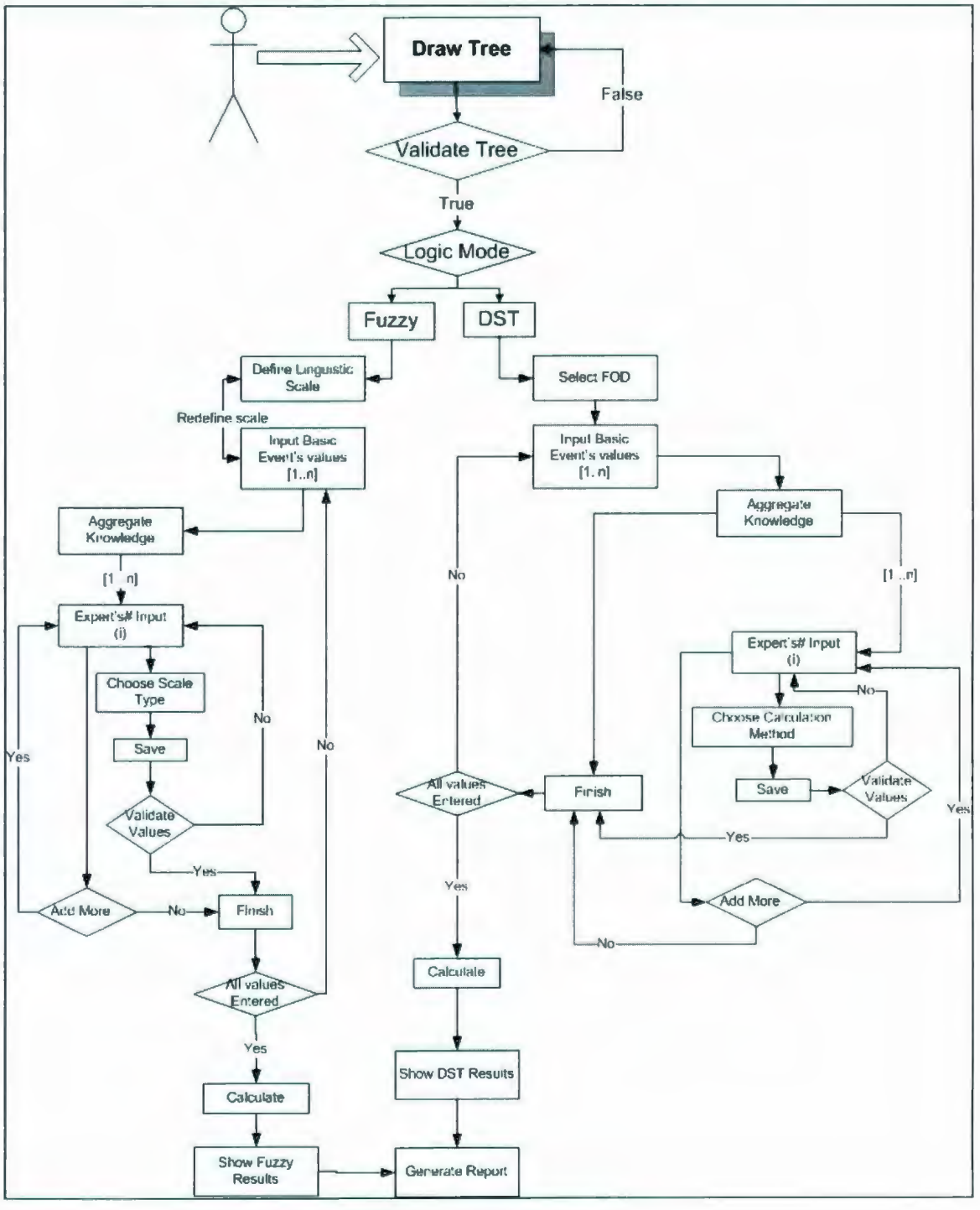


Figure 3.8: Work flow diagram of the ***ExpertFTA***

3.4.2 Design Patterns Used in *ExpertFTA*

ExpertFTA has been designed by implementing multiple design patterns to handle its complex functionalities. These include ***Factory Method*** (Creational Pattern), ***Adapter*** (Structural Pattern), ***Command*** and ***Mediator*** (Behavioural Pattern). Two of them (*Command, Mediator*) are briefly described below:

In *Mediator* design pattern, communications between objects are encapsulated within the Mediator object. Objects in the entire system interact with each other through the Mediator. The relationship between the control class and other participating classes is multi-directional (Gamma, 1995). By using this design pattern, ***ExpertFTA*** gains a centralized control to manipulate participating objects, simplifies the protocol, improve objects reusability and lowers the coupling. Figure 3.9 depicts the component model of all modules in ***ExpertFTA*** which uses the Mediator design pattern.

The GUI module in figure 3.9 needs to be addressed seperately to illustrates its design. Figure 3.10 depicts the component model of the GUI module. This module of ***ExpertFTA*** has adopted the concept of Blackboard architecture. The Blackboard architecture model is a distributed computing architecture where a common data structure is shared by multiple agents/systems.

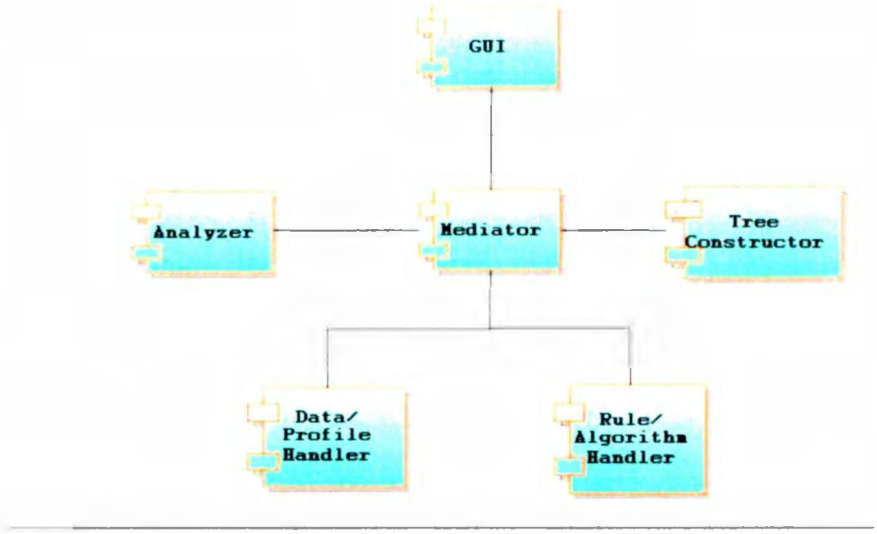


Figure 3.9: Component Model of ***ExpertFTA***

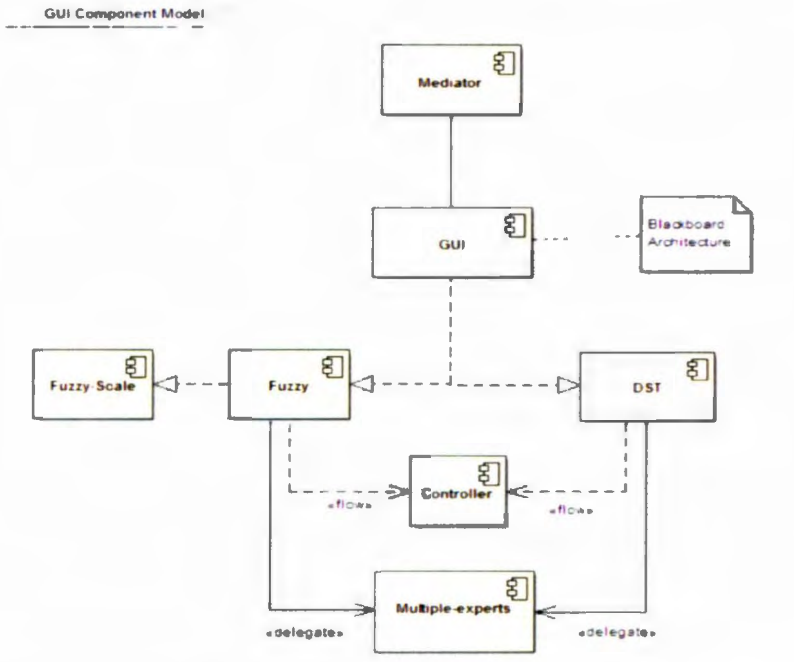


Figure 3.10: Component Model of GUI

In the GUI module the following information or metadata is stored in memory:

- Number of levels in the fault tree.
- Number of items (gate or basic event) in a particular level.
- Type of calculation (DST or Fuzzy).
- User defined Fuzzy scale.
- Multiple experts' knowledge for each basic event.

This tool allows users to undo or remove any gate/basic event added to the Fault-Tree. In order to support the undoing of actions, the *Command Pattern*, which treats actions as objects (Jia, 2003), has been implemented. The intention of this design pattern is to encapsulate any action of the ***ExpertFTA*** as an object. The participants of the Command design pattern are the following:

- *Command* (e.g., Command, undoableCommand), which defines an interface to perform or undo an action.
- *Receiver* (e.g., Analyzer), which knows how to perform actions.
- *ConcreteCommand* (e.g., CommandAnalyzer), which implements the *Command* interface and delegates the execution of the action to the *Receiver.*
- *Client* (e.g., fdftaGUI.KeyListener), which creates the concrete commands and binds the concrete commands to their receivers.
- *Invoker* (e.g., Analyzer), which asks the command to carry out the action.

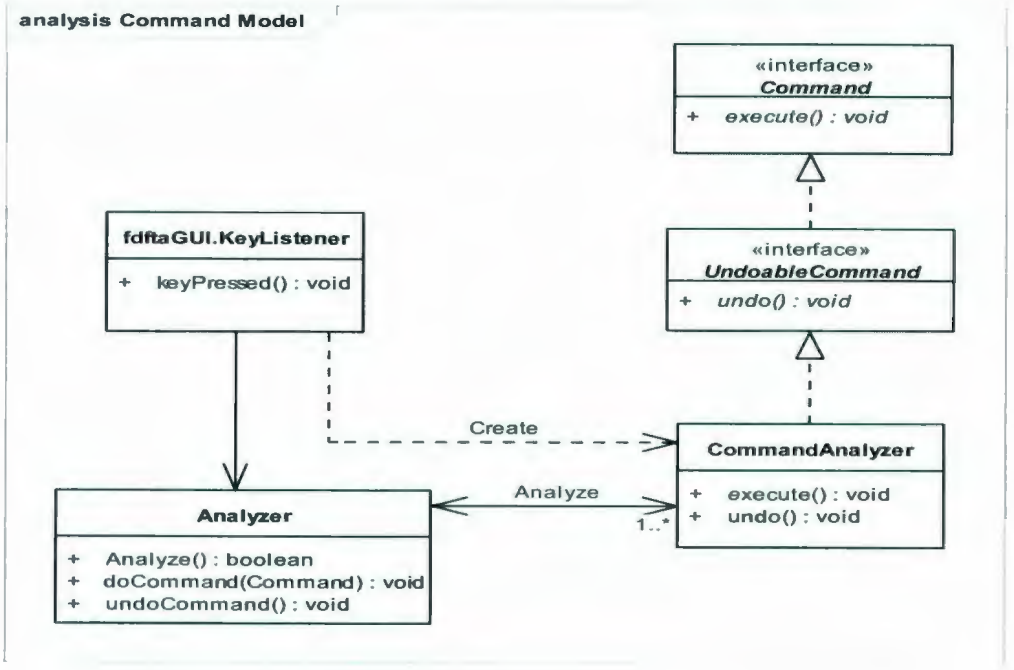


Figure 3.11: Command model analysis

3.5 Summary

This chapter has provided the description of the ***ExpertFTA*** from the design and modelling point of view. Some of the important lessons learned are as follows:

- While designing a complex software system, it is crucial to pay attention to the details of the requirements. By observing those requirements, workflow diagrams and wireframes of possible screens are developed. These are the information which can be crucial as well as informative for further development.

- Selecting the right design pattern(s) is a major task to develop a complex tool. By identifying appropriate design pattern(s), one can get clear overview of the entire development phase.

Chapter 4 ExpertFTA SYSTEM DEVELOPMENT

This chapter concentrates on describing the overall architecture as well as the development of the ***ExpertFTA***.

4.1 *ExpertFTA* Architecture Overview

As already discussed in earlier chapters, it is very important to make sure that all objects and modules in software have low coupling and high cohesion. By keeping that concept in mind, ***ExpertFTA*** is architected in such a fashion so that it reflects that in all possible areas.

Figure 4.1 shows an overview of the architecture of the ***ExpertFTA*** from different module point of view. In that figure, each module shows the different functionalities that it is responsible for. For example, "GUI" module contains all the functionalities to deal with the interaction with the fault tree. It should be noted that the "Draw and construct Fault Tree" is not the name of an object; rather it is the description of the functionality that belongs to this particular module. As Figure 4.1 depicts that the Mediator module is the core of the architecture which also includes the Command design pattern. The rest of the modules are independent from each other and communicates through the Mediator module. In this way modifying any of the modules will not affect the others. Below each module are discussed from the functionalities point of view.

The "GUI" module creates and facilitates the user with a canvas to draw and modify the fault tree. Creating a gate or BE (basic event) object is done from this module. Though this

module handles knowledge aggregation and constructing Fuzzy Linguistic Scale, these two functionalities are defined as separate entity or module.

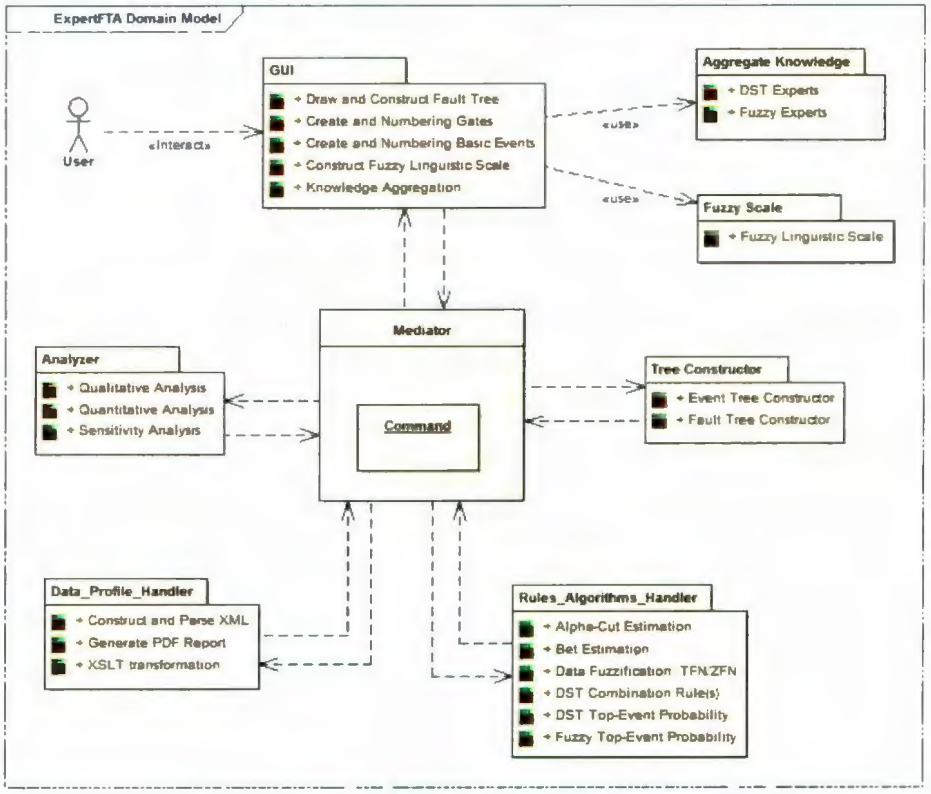


Figure 4.1: ***ExpertFTA*** architecture overview

The "Analyzer" module contains all of the functionalities associated with analysis, such as qualitative, quantitative and sensitivity analysis.

ExpertFTA is architected such that in future ETA (Event Tree Analysis) also can be implemented for both Fuzzy and DST analysis. Some of the core functionalities of the ETA are already developed and implemented which can further be enhanced to make it fully functional. "Tree Constructor" module depicts those functionalities.

The "Data_Profile_Handler" module provides the functionalities that is required for this tool to a provide data profiling mechanism. These include handling XML files and generate PDFs by rendering XSLT.

The "Rules_Algorithm_Handlers" is the module which is basically responsible for all sorts of calculations including alpha-cut estimation, Bet estimation, data fuzzyfication and top event probability calculation.

4.2 *ExpertFTA* Functionalities Overview

At the very first screen of constructing a Fault Tree, the user can draw his/her desired fault tree by just clicking on the appropriate gates/ basic event symbol. Upon finishing drawing the tree, the user selects an option from "DST" or "Fuzzy" as a logic mode for calculation. Based on the selected option, the application gathers necessary information as the user proceeds with inserting values. By choosing "DST", the user gets a screen where he/she can insert multiple experts' knowledge for a particular basic event. By choosing "Fuzzy" as a logic mode, the user is able to define his/her own fuzzy linguistic scale along with the above knowledge aggregation option for a single basic event. And the "Most Likely Value" scale in this logic mode will convert the entered value into a fuzzy data set by implementing a triangular/trapezoidal function.

As the user adds or connects gates and basic events with the fault tree, this tool will create an XML file behind the scene. From reusable aspect, XML is ideal because the user can easily refer to the previous file and also modify that file in future if needed. This file also becomes handy when it needs to be analyzed or port to other systems or environments.

Once the calculation is finished, the *'Analysis'* tab is displayed for the selected logic mode. In both cases, the *'Analysis'* tab contains the gate-by-gate results for the entire fault-tree. In the fuzzy-analysis tab, in addition to that result, the user can also be able to perform an alpha-cut calculation. In the DST- analysis tab, the result consists of "Bet" estimation for each gate. Finally, the user can generate a PDF-report which captures the values of each gate, basic events and also all information gathered from experts.

4.3 Basic Tree Generation of the Fault-Tree

A Fault-tree consists of a top-event, basic events and logic-gates. Each of them is treated as a node and represented by an object. Two kinds of gates, **AND** and **OR,** are sub-classes of logic-gates with their own characteristics. Each basic event is also be defined as an object. The advantage of encapsulation (hiding the implementation details from object point of view) and polymorphism (that allows different functionalities to be handled using a uniform interface) from object has acquired here. Figure 4.2 illustrates the object orientation for different events in ***ExpertFTA***.

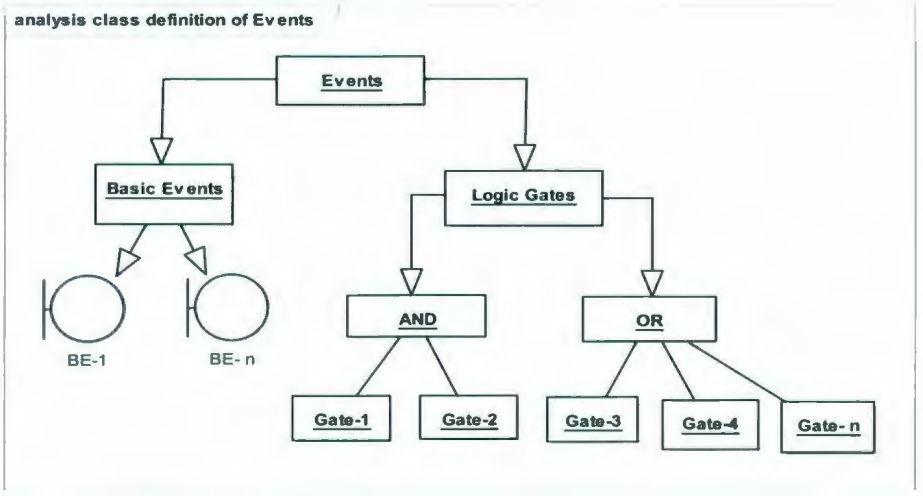


Figure 4.2: Different types of events and gates

The graphical user interface of ***ExpertFTA*** enables users to draw a fault tree by inserting, deleting or adding events. *Canvas* is the area on the screen where the fault tree gets drawn by the user. In order to draw a tree and add/remove any events from it, ***ExpertFTA*** divides the canvas area into *'Level'*s. Every single node of the tree falls into a specific *Level*. This tool maintains a collection of these *Level* structures and gets automatically updated upon any modification in the fault tree. At the beginning of the program, co-ordinates of the canvas area get evaluated based on the users' screen size. For every node in the tree, the drawing-area co-ordinates of that node get calculated and are stored into the object. While the tree gets updated, the program re-evaluates each node's drawing-area co-ordinates and *Level* structure recursively. An individual node or object has knowledge about its own drawing-area co-ordinates and *Level* structure that it belongs. If any new gate or basic event gets added in any level of the tree, the program assigns an appropriate *id* to it based on the previous event's *id* of the same type. Upon deleting an event, all events of that type gets re-evaluated in order to re-assign the new *id*.

4.4 Object Orientation and Control Flow

Once the user finishes drawing the desired fault tree and clicks the *"Finish"* button to proceed further, the application creates objects for each symbols in that fault tree. After that a tree validation is performed in order to verify that all gate objects have at least two children events connected with it. Then the application creates (or refreshes if it is a modification in the existing tree) a XML file. This XML file is initialized with attribute nodes for the selected *Logic-Mode* ("Fuzzy" or "DST") and the current system time. An XML node gets created and appended accordingly for each object created by the application. The *Gate* node contains *id* (e.g. "GATE_1"), *gateType* ("AND" or "OR"), *value* (calculated value which is empty initially) and *Bet* (Bet estimation) attribute-nodes along with child node(s) for any child object(s) coupled with this gate. The *BasicEvent* node also contains *id* (e.g. "BE_1") and *value* (combination of expert's knowledge value which is empty initially) attribute-nodes. Any expert's knowledge gathered for this particular basic event gets added as a child node (named as *ExpertOpinion*) of this *BasicEvent*- node. An *id* (e.g. "Expert_1") and *value* (value entered by the user for this expert) are two attribute-nodes associated with the *ExpertOpinion*-node.

Table 4.1 gives a brief idea about the signature of classes and methods that describes here. *FaultTreeGate* is an object which represents a gate of the fault tree. This object maintains a collection of child(ren) associated with it. The method "*calculateResult(...)*" in this object is responsible for invoking "*calculate(...)*" method on a specific logical operation and pass the current child-collection to it. This object also contains a method which is responsible for calculating Bet (calls *'calculateBet()'*) based on the logic-mode. There are two objects to handle logical operations: '*AndOperation*' and '*OrOperation*'. Both objects iterate through the child list and recursively traverse the entire tree to calculate the final result. The property

FaultTreeGate.isVisited is used in order to avoid unnecessary multiple access of a certain gate during tree traversal.

Table 4.1: Constructor and method signature of different events

Java Class	***Constructor's Signature***	***Method's Signature***
FaultTreeGate	*FaultTreeGate(String gateType, String gateName, FaultTreeGate parent);*	*double[] calculateResult (String logicMode);* *void calculateBet(String logicMode);* *boolean isVisited();*
AndOperation	*AndOperation(List<BasicEventInFTA> basicEvents);*	*static BigDecimal[] calculate(List<FtaItem> ftElements, String dst_fuzzy);*
OrOperation	*OrOperation(List<BasicEventInFTA> basicEvents);*	*static BigDecimal[] calculate(List<FtaItem> ftElements, String dst_fuzzy);*
FuzzyProbability Convertor	*FuzzyProbabilityConvertor()*	*trapizoialConversion() ;* *triangularConversion();*
BasicEventInFTA	*BasicEventInFTA(String eventID, String parentID)*	*combineExpertKnowledges()*

Selecting a logic-mode (Fuzzy/DST) before clicking the *"Finish"* button determines which object will be instantiated for the next phase. *'FuzzyExpertInputPanel'* and *'DSTInputPanel'* are the two objects that are responsible to interact with the user to get input in the next step. If either of these objects are already instantiated, then it gets updated by the latest change in the fault tree. *FuzzyExpertInputPanel* permits the user to construct a new fuzzy linguistic scale to use during the current session. By clicking the "*Define*" button the user gets a screen where number of category in the scale including the suitable linguistic variable name can be assigned. Also upon selecting the membership function (TFN: triangular fuzzy number or ZFN: trapezoidal fuzzy number), the same screen provides an option to assign corresponding fuzzy values (e.g. for TFN: Lower, Most likely and upper bound) for the particular user defined category. Once the linguistic

scale is saved, this is available for assigning a fuzzy value in the current session. In both logic-modes, for each *BE,* knowledge aggregation has to be performed at least with one expert's opinion. *DSTInputPanel* enables the user to choose from FOD type ("*true/false*" or "*success/failure*") during individual basic event's knowledge aggregation and calculation method (DS/Yager) for each expert's opinion. Upon clicking the "*Calculate*" button on the screen, data validation is performed by checking completeness and consistency of values for all the *BEs*. And also invokes appropriate procedures to calculate top event's probability and show the gate-by-gate results on the next tab. *'AnalysisPanel'* is the object which is responsible to show the results and the alpha-cut calculation sub-panel based on the selected logic-mode. In the alpha-cut calculation section, the user can check the results for any entered value in the alpha-cut input box. Though the calculation for ZFN has been implemented, it is not yet usable from GUI.

4.5 Converting Probability Data into Fuzzy or DST Data

Fuzzy probability uses the fuzzy number, which is expressed by a fuzzy set and characterized by its membership function, μ. It can be represented by a triangular or trapezoidal shape or bell shaped membership function (Cheng et al., 2000).

A java class, *FuzzyProbabilityConvertor,* fuzzify the provided data for *BasicEventInFTA*. The method- *trapizoialConversion()* and *triangularConversion()* of that class is responsible for converting data. Table 4.2 and 4.3 represents the implementation strategies for TFN and ZFN respectively:

Table 4.2: Implementation strategies for TFN

Strategy 1: ***$0 \leq M \leq 0.5$***	***Strategy 2:*** ***$0.5 \leq M \leq 1.0$***
Lower bound value $P_L = M \times 0.5$ Upper bound value $P_U = M \times 1.5$	Lower bound value $P_L = (3*M - 1)/2$ Upper bound value $P_U = (M + 1)/2$

Based on the given "most likely value" the application will generate a lower bound and an upper bound value by implementing logic stated table 4.2. In that table, "M" stands for "Most likely value".

Table 4.3: Trapezoidal conversion strategies

Strategy 1: ***$0 \leq \beta \leq 0.5$***	***Strategy 2:*** ***$0.5 \leq \beta \leq 1.0$***
Lower bound value $P_{iA} = \beta \times 0.5$ Lower bound value $P_{iB} = \beta \times 0.75$ Upper bound value $P_{iC} = \beta \times 1.25$ Upper bound value $P_{iD} = \beta \times 1.5$	$\Delta = (1 - \beta)/4$ Lower bound value $P_{iA} = (\alpha - (2 \times \Delta))$ Lower bound value $P_{iB} = (\alpha - \Delta)$ Upper bound value $P_{iC} = (\alpha + \Delta)$ Upper bound value $P_{iD} = (\alpha + 2 \times \Delta)$

Note: β = Probability value, α = Fuzzy Number

4.6 Knowledge Aggregation for DST Using Matrix Computation

In DST, for multi-experts' knowledge aggregation, the combination rule requires matrix computation in order to merge the knowledge into a belief structure. A number of matrices can be generated based on the number of distinct combinations of experts/sources. For example, if user gathers knowledge from three independent sources (m_1, m_2, m_3), there will be three matrices $m_1 \oplus m_2$, $m_1 \oplus m_3$ and $m_2 \oplus m_3$. The following algorithm is developed to generate the number of possible combination from the given set of expert knowledge:

```
for i = 1 to n-1
    for j = i+1 to n
        m_i ⊕ m_j
    end
end
```

Each element in the combination is treated as a column-vector and row-vector respectively and the computed matrix is the result of the outer product of the combination. This square *(n x n)* matrix's *'n'* is equal to the number of the subset of the power set *(P)*. By neglecting the null set, we have total 2^n -1 number of rows (and columns).

The java class- *BasicEventInFTA* contains a method- *combineExpertKnowledges()* that deals with the above operations. This method returns an object of type *ExpertsOpinion*. *ExpertsOpinion* is an interface implemented by two classes, *SingleExpert* and *CombineTwoExperts*. The class *CombineTwoExperts* requires two experts knowledge, which need to be passed as arguments during object-construction. Table 4.4 shows the pseudo-code of *combineExpertKnowledges()*.

Table 4.4: Pseudo code of knowledge combination

```
+ combineExpertKnowledges(): ExpertsOpinion
  If SingleExpert then
     ExpertsOpinion → SingleExpert;
  Else: // for multiple experts
        For number of experts
              ExpertsOpinion = CombineTwoExperts(Expert_i, Expert_j);
              Expert_j → ExpertsOpinion;
        End of for;
  End of if;
  Return ExpertsOpinion;
```

4.6.1 Intersection matrix:

Intersection Matrix (*IM*) shows the intersection of each subset of power set *(P)* with one another. For example, if Ω = {T, F} *(where, T = "True" and F = "False")*, then that leads to four subsets, i.e., {ϕ (a null set), {T}, {F}, {T, F}} in power set (*P*). And by ignoring the null set, we have 3x3 matrix as in Table 4.5:

Table 4.5: Matrix representation of Ω

	{T}	*{F}*	*{T,F}*
{T}	T	0	T
{F}	0	F	F
{T,F}	T	F	T,F

From the matrix in Table 4.5, the intersection value(s) for *'T'*, *'F'*, *'T,F' and 'Conflicts'* are identified accordingly. Those distinct positions can be stored in one-dimensional array:

$$T = [\, IM(1,1),\ IM(1,3),\ IM(3,1)] \quad (1)$$

$$F = [\, IM(2,2),\ IM(2,3),\ IM(3,2)] \quad (2)$$

$$T,F = [\, IM(3,3)] \quad (3)$$

$$\text{Conflicts} = [\, IM(1,2),\ IM(2,1)] \quad (4)$$

The above collections are handy while computing result by using the selected combination rule. The Java class- *CombineTwoExperts,* computes a matrix by outer product of the given data and extracts the values from the resultant matrix using the knowledge of the above predefined collections.

4.7 User view of *ExpertFTA*

A presentation of the users view and workflow is outlined below. Figure 4.3 depicts the very first screen where the user can draw the fault tree by clicking the symbols from the symbol palette which is on the left side of the screen. Symbols which are not currently available are greyed out. By double-clicking on the specific gate in the fault tree, the user can select that gate and a single mouse click on the particular symbol from the symbol palette appends it with the selected gate. The application allows the user to delete any selected *BE* or gate (and any sub-tree underneath it) from the tree by just clicking the delete icon. As the fault tree is extended, it may become too large to be viewed on the canvas. Horizontal and vertical scrollbars of the canvas can be used to navigate around in order to view the hidden areas of the tree. The user can also use the *'zoom in'* and *'zoom out'* icon in order to increase and decrease the zoom value of the entire fault tree on the canvas respectively.

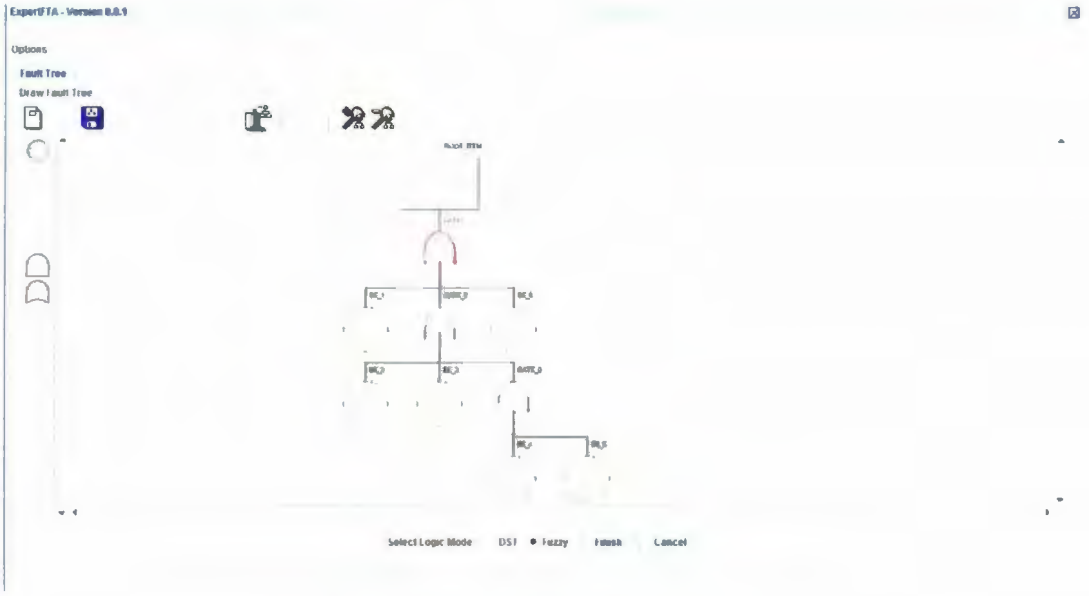


Figure 4.3: A screenshot of ***ExpertFTA***

4.8 Fuzzy Input Parameter screen

The "Fuzzy Input Parameter" tab enables the user to view the one-to-one relation with each gate and *BE*. Figure 4.5 shows the screenshot of this tab. One of the key features of this tool is defining users' own linguistic scale for Fuzzy input. Before inserting value for *BE*, the user needs to construct the linguistic scale by clicking the "Define" button.

Figure 4.4: A screenshot of "Fuzzy Input parameter"

As the user clicks the "Define" button, the program prompts a separate screen (Figure 4.6) to construct the linguistic scale. Once the user finishes creating the scale by clicking the *"Save"* button, the application stores it in memory for later use. The *"Expert's knowledge..."* button enables a detailed input screen for that particular *BE*.

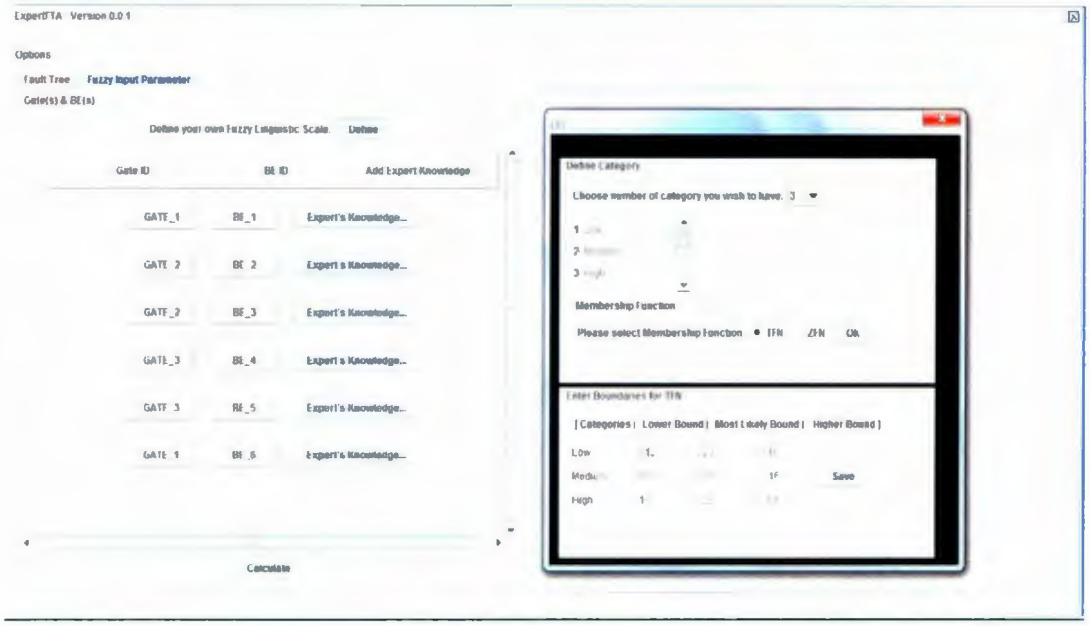


Figure 4.5: A screenshot of "Fuzzy Linguistic scale" screen

Figure 4.6 represents the detail input screen for **BE_1**. User can choose different scale type from the combo-box in order to assign value for a certain "Expert". Currently available scales are: *"Linguistic Scale"*, *"Most Likely Value"* and *"Crisp"*. Choosing *"Linguistic Scale"* provides the fuzzy variables which have been defined earlier and selecting any one from that list populates the placeholder for "P1", "P2" and "P3". "P1", "P2" and "P3" essentially stands for "Lower", "Most likely" and "Higher" bound.

While *"Most Likely value"* scale prompts an input-box (as shows on the Figure 4.6) to enter error-factor value from which "P1" and "P3" gets calculated accordingly, *"Crisp"* scale enables "P1" text field to enter the value. The user has to click "*Save*" button to store the input values for the specific "Expert" and "Add..." button to add another "Expert" for the current *BE*.

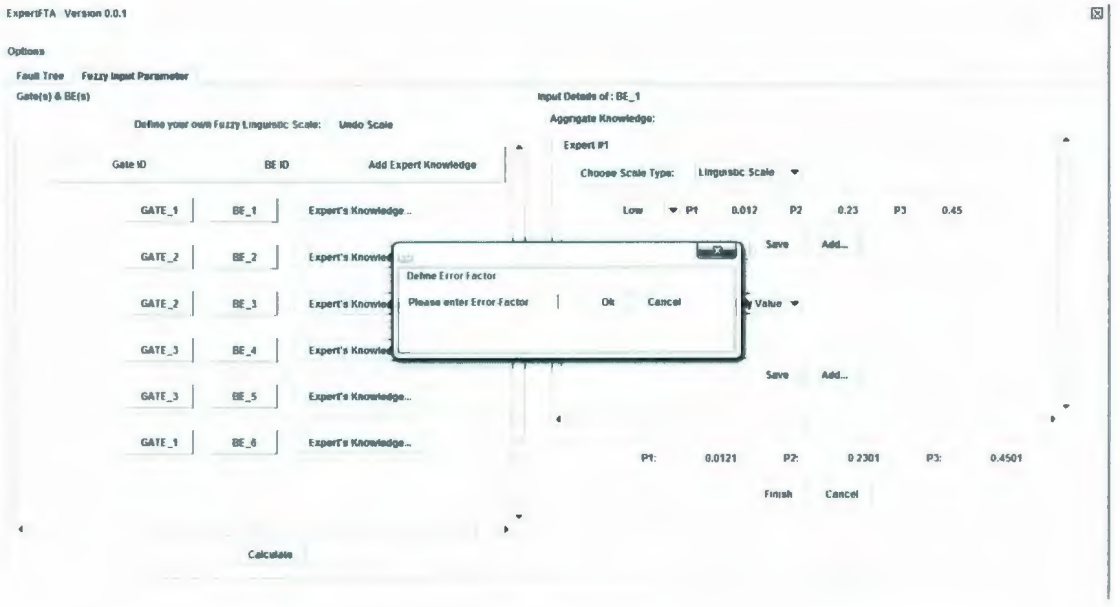


Figure 4.6: A screenshot of "Input Detail" screen

On the bottom of the input-detail screen the total average "P1", "P2" and "P3" is shown for the particular *BE* (as shown in Figure 4.6). The user has to click the "*Finish*" button in order to tell the program that entering values for the *BE* has completed. Once all *BEs* values are entered, upon clicking the "*Calculate*" button the user validates all *BEs* and calculates the gate-by-gate probability.

The "Fuzzy Analysis" screen (as shown in figure 4.7) shows the gate wise result on the left hand side of the screen. And alpha-cut calculation can be made on the right hand side of the screen.

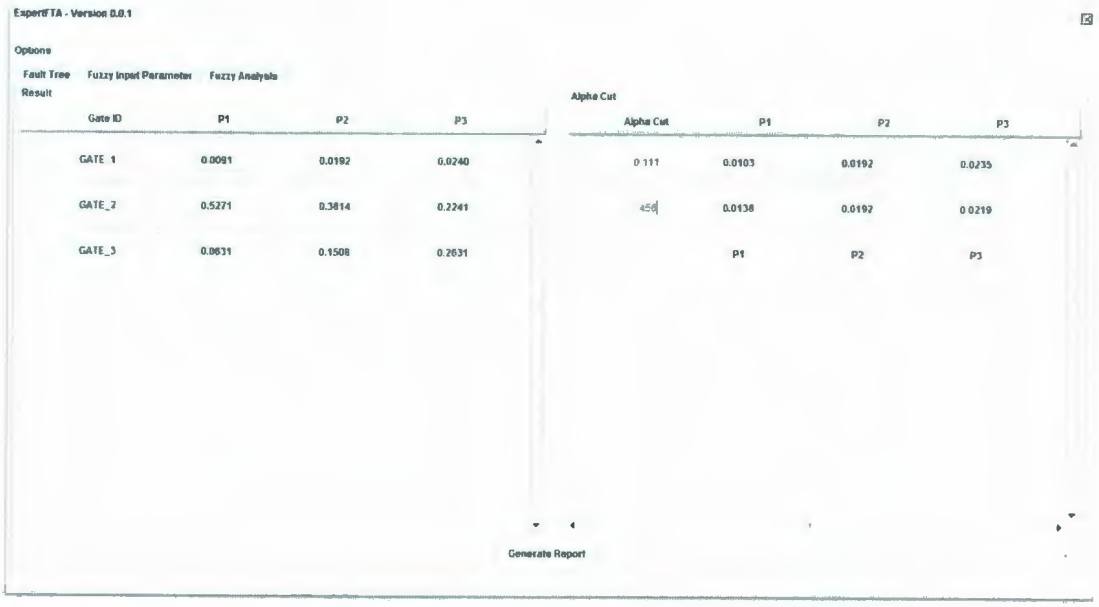
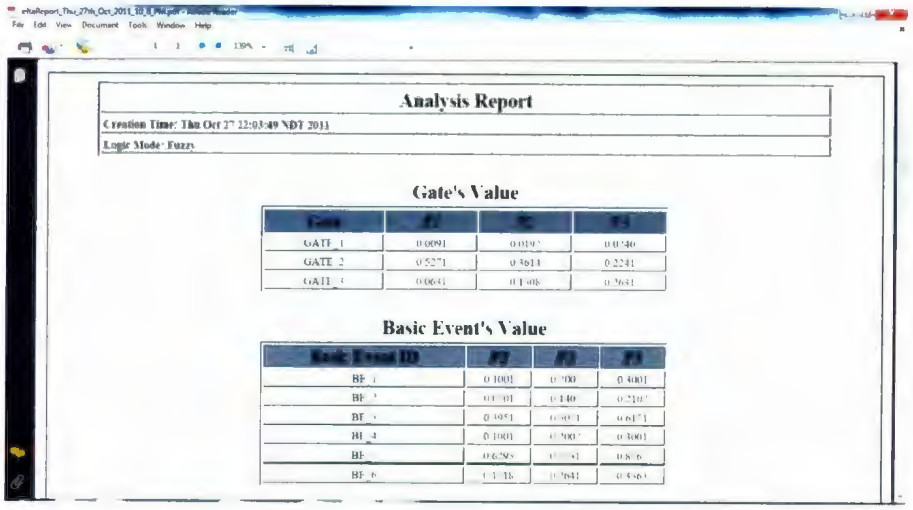


Figure 4.7: A screenshot of "Fuzzy Analysis" screen

When the user clicks the "*Generate Report*", a PDF (Portable Document Format) file is created with the date and time stamped on it. Figure 4.8(a) and figure 4.8(b) shows the generated report from the example in Figure 4.3.



Analysis Report

Creation Time: Thu Oct 27 22:03:49 NDT 2011

Logic Mode: Fuzzy

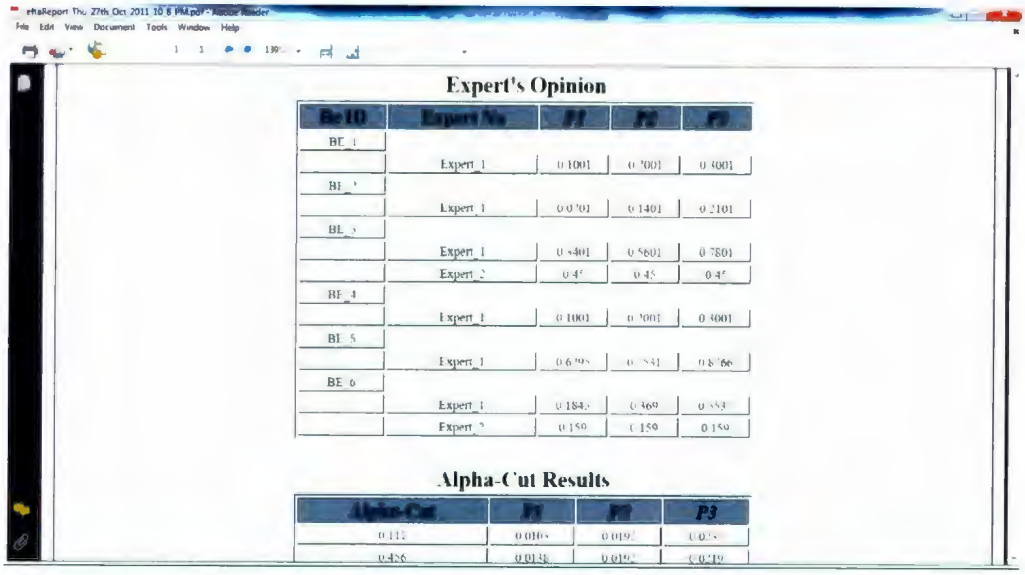
Gate's Value

Gate	P1	P2	P3
GATE_1	0.0091	0.0192	0.0340
GATE_2	0.5271	0.3614	0.2241
GATE_3	0.0643	0.1406	0.2631

Basic Event's Value

Basic Event ID	P1	P2	P3
BE_1	0.1001	0.2001	0.4001
BE_2	0.0701	0.1401	0.2101
BE_3	0.3951	0.4951	0.6171
BE_4	0.1001	0.2001	0.3001
BE_5	0.6295	0.7541	0.8766
BE_6	0.1176	0.2641	0.4563

Figure 4.8(a): A screenshot of fuzzy report



Expert's Opinion

Be ID	Expert No	P1	P2	P3
BE_1				
	Expert_1	0.1001	0.2001	0.4001
BE_2				
	Expert_1	0.0701	0.1401	0.2101
BE_3				
	Expert_1	0.3401	0.5601	0.7801
	Expert_2	0.45	0.45	0.45
BE_4				
	Expert_1	0.1001	0.2001	0.4001
BE_5				
	Expert_1	0.6295	0.7541	0.8766
BE_6				
	Expert_1	0.1845	0.369	0.553
	Expert_2	0.159	0.159	0.159

Alpha-Cut Results

Alpha-Cut	P1	P2	P3
0.112	0.0105	0.0192	0.0276
0.426	0.0138	0.0192	0.0219

Figure 4.8(b): A screenshot of fuzzy report (continued…)

ExpertFTA generated the following XML file which has been parsed by the XSLT in order to create report:

Table 4.6: Example of the XML file

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <FaultTree LogicMode="Fuzzy" curTime="Thu Oct 27 22:03:49 NDT 2011">
- <Gate Bet="" ID="ROOT_ITEM" gateType="ROOT_ITEM" value="">
- <Gate Bet="0.0142" ID="GATE_1" gateType="AND" value="0.0091, 0.0192, 0.0240">
- <BasicEvent ID="BE_1" value="0.1001, 0.2002, 0.3001">
<ExpertOpinion ID="Expert_1" value="0.1001, 0.2001, 0.3001" />
</BasicEvent>
- <Gate Bet="0.4443" ID="GATE_2" gateType="OR" value="0.5271, 0.3614, 0.2241">
- <BasicEvent ID="BE_2" value="0.0701, 0.1402, 0.2102">
<ExpertOpinion ID="Expert_1" value="0.0701, 0.1401, 0.2101" />
</BasicEvent>
- <BasicEvent ID="BE_3" value="0.3951, 0.5051, 0.6151">
<ExpertOpinion ID="Expert_1" value="0.3401, 0.5601, 0.7801" />
<ExpertOpinion ID="Expert_2" value="0.45, 0.45, 0.45" />
</BasicEvent>
- <Gate Bet="0.1070" ID="GATE_3" gateType="AND" value="0.0631, 0.1508, 0.2631">
- <BasicEvent ID="BE_4" value="0.1001, 0.2002, 0.3001">
<ExpertOpinion ID="Expert_1" value="0.1001, 0.2001, 0.3001" />
</BasicEvent>
- <BasicEvent ID="BE_5" value="0.6295, 0.7531, 0.8767">
<ExpertOpinion ID="Expert_1" value="0.6295, 0.7531, 0.8766" />
</BasicEvent>
</Gate>
</Gate>
- <BasicEvent ID="BE_6" value="0.1718, 0.2641, 0.3563">
<ExpertOpinion ID="Expert_1" value="0.1845, 0.369, 0.5535" />
<ExpertOpinion ID="Expert_2" value="0.159, 0.159, 0.159" />
</BasicEvent>
</Gate>
</Gate>
- <Alpha-Cut_Value ID="Alpha-Cut_Value">
<Alpha-Cut ID="0" value="0.111, 0.0103, 0.0192, 0.0235" />
<Alpha-Cut ID="1" value="0.456, 0.0138, 0.0192, 0.0219" />
</Alpha-Cut_Value>
</FaultTree>
```

4.9 DST Input Parameter Screen

Similar to the "Fuzzy Input Parameter" screen, for DST, the program creates a "DST Input Parameter" tab which depicts the one-to-one relation with each gate and *BE*. Figure 4.9 shows the screenshot of "DST Input Parameter" tab.

Figure 4.9: A screenshot of "DST Input parameter"

Upon clicking the *"Expert Knowledge..."* button for specific a *BE*, the user gets the input details screen for that *BE*. Initially the user needs to choose the type of FOD. Currently two types of FOD ({True/False}, {Success/Failure}) are predefined by the application which can be selected from the combo box. By choosing an appropriate FOD, the user can apply it for every

expert's knowledge aggregation for the particular *BE*. For each expert's knowledge aggregation, the user has to choose the combination method ('DS'/ 'Yager') prior to clicking the "*Save*" button to store the entered value. Once the "*Save*" button is clicked, it becomes disabled, and if the user makes any change in the input values, the program re-enables this button. Same as Fuzzy input detail screen, the "*Add...*" button appends another row for the next expert. At the bottom of the input detail screen, the total calculated "Belief" and "Plausibility" for the entered values are shown. "*Finish*" and "*Calculate*" button have similar functionalities as Fuzzy input detail screen.

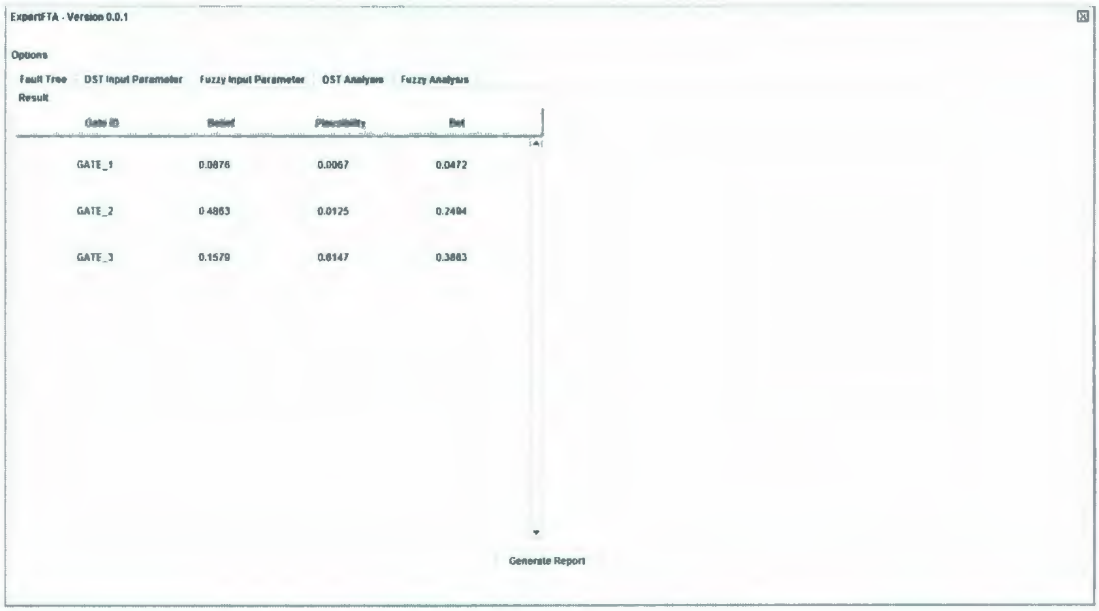


Figure 4.10: A screenshot of "DST Analysis"

As Figure 4.10 depicts that the "*DST Analysis*" tab show the gate-by-gate results of "Belief", "Plausibility" and "Bet" calculations accordingly. The "*Generate Report*" button creates a similar PDF report as described for the "Fuzzy Analysis" screen (section 4.8).

0 Summary

This chapter has provided the description of the ***ExpertFTA*** from the architectural point of view. This also described the implementation details of some major elements in this tool. Later in this chapter, user interface of the ***ExpertFTA*** has elaborated to show the functionalities of this tool.

Chapter 5 CASE STUDY AND RESULT

This chapter discusses the application of the ***ExpertFTA*** using a case study. Result from the case study is also provided.

5.1 Case Study

An earlier published real life study has been adopted here. The study is published by Sadiq *et al*, (2008) in "Predicting risk of water quality failures in distribution networks under uncertainties using fault-tree analysis". Here Sadiq *et el.* investigated two studies of water quality failure in Canada: North Battleford (Saskatchewan, in April 2001) and Walkerton Ontario(in May 2000). To verify the applicability of ***ExpertFTA***, it is decided to choose the case of Walkerton Ontario.

5.1.1 Subjective (Fuzzy-Based) Approach

To demonstrate the subjective (Fuzzy-based) approach feature of ***ExpertFTA,*** the probability of basic events are defined using the linguistic expression and TFN values which are stated in Table-5.1. These values have been taken from the same case study (Sadiq *et el.*, 2008) and used to develop a fuzzy scale for the case study using *ExpertFTA* and also to calculate the TFN of the top-event.

Table 5.1: Fuzzy linguistic scale

ID	*Likelihood*	*Triangular Fuzzy Number(TFN)*
1	Absolutely low	(0, 0, 0.1)
2	Extremely low	(0, 0.1, 0.2)
3	Quite low	(0.1, 0.2, 0.3)
4	Low	(0.2, 0.3, 0.4)
5	Mildly low	(0.3, 0.4, 0.5)
6	Medium	(0.4, 0.5, 0.6)
7	Mildly High	(0.5, 0.6, 0.7)
8	High	(0.6, 0.7, 0.8)
9	Quite High	(0.7, 0.8, 0.9)
10	Extremely High	(0.8, 0.9, 1.0)
11	Absolutely High	(0.9, 1.0, 1.0)

Table 5.2 depicts the Names, Descriptions and Values considered for Basic Events for the case study. Values for "Expert-2" and "Expert-3" have been considered randomly whereas "Expert-1" values are adopted from Sadiq *et el.,* (2008).

Table 5.2: Name, Description and Value assumed for Basic Events

Basic Event ID	*Name*	*Description*	*Expert 1*	*Expert 2*	*Expert 3*
BE_1	Source water	Raw water contamination potential	Mildly low	Low	Medium
BE_2	Treatment units	Conditions of the treatment units in the plant before water enters the distribution system.	Mildly High	Medium	Mildly low
BE_3	Untrained staff	Lack of knowledge, experience and education	Mildly High	Mildly low	Medium
BE_4	Broken pipes & gaskets	Risk of water quality failure due to broken pipes and gaskets throughout the water distribution system	Low	Quite low	Mildly low
BE_5	Maintenance and repair events	Risk of water quality failure due to the intrusion of contaminants during maintenance and repair of	Low	Quite low	Mildly low

		pipes and other components of the distribution system.			
BE_6	Cross-connections	Risk of water quality failure due to the intrusion of contaminants at the cross-connections of the distribution system.	Extremely low	Absolutely low	Absolutely low
BE_7	Loss of pressure	Likelihood of pressure loss	Extremely low	Absolutely low	Absolutely low
BE_8	Presence of contaminants	Likelihood of contaminants' presence	Low	Quite low	Mildly low
BE_9, BE_11	Metallic surface	Risk of water quality failure due to leaching in the absence of a metallic surface.	Extremely High	Absolutely High	Extremely High
BE_10	Leaching conducive conds.	Conditions conducive to leaching leading to risk of water quality failure.	Extremely low	Absolutely low	Absolutely low
BE_12	Corrosion conducive conds.	Risk of water quality failure due to corrosion likelihood of conditions conducive to corrosion.	Quite low	Extremely low	Low
BE_13	Threat	Direct and indirect threats to the distribution system	Extremely low	Absolutely low	Absolutely low
BE_14	Physical & Geo. vulnerability	Risk of water quality failure due to the surrounding physical and geographical vulnerability of the system.	Extremely low	Absolutely low	Absolutely low
BE_15	Insufficient monitoring	Risk of water quality failure due to insufficient monitoring in a distribution system.	Low	Quite low	Mildly low
BE_16, BE_19	Organic matter	Risk of water quality failure due to DBP formation likelihood of organic matter	Extremely low	Absolutely low	Absolutely low
BE_17, BE_20	DBP conducive conds.	Risk of water quality failure due to DBP	Quite low	Extremely low	Low

		formation under likelihood of conducive conditions			
BE_18	Concentration of residual disinfectant	Risk of water quality failure related to disinfectant loss under likelihood of residual disinfectant	Quite low	Extremely low	Low
BE_21	Disinfectant loss conducive conds.	Risk of water quality failure related to disinfectant loss under likelihood of conducive conditions	Quite low	Extremely low	Low
BE_22	Plastic	Plastic material present or not	Absolutely low	Extremely low	Quite low
BE_23	Hydro carbon	Hydro carbons are present or not around the pipe	Absolutely low	Extremely low	Quite low
BE_24	Permeation conducive conds.	Exposure of hydrocarbons to plastic material over long time	Absolutely low	Extremely low	Quite low
BE_25	Organic matter	Presence of optimal organic matter for the growth of biofilm	Extremely low	Absolutely low	Absolutely low
BE_26	Detention time	The time water stays in the system (water age)	Quite low	Extremely low	Absolutely low
BE_27	Biofilm conducive conditions	Temperature, nutrients etc.	Quite low	Extremely low	Low

5.1.2 Evidence Theory-Based Approach

To demonstrate the evidence theory-based approach feature of this tool, a case study was also performed using the same fault tree. Figure-5.1 shows the screenshot of the entire fault tree (zoomed-in to fit on the screen) which has been drawn for DST (Dempster–Shafer Theory of Evidence) analysis.

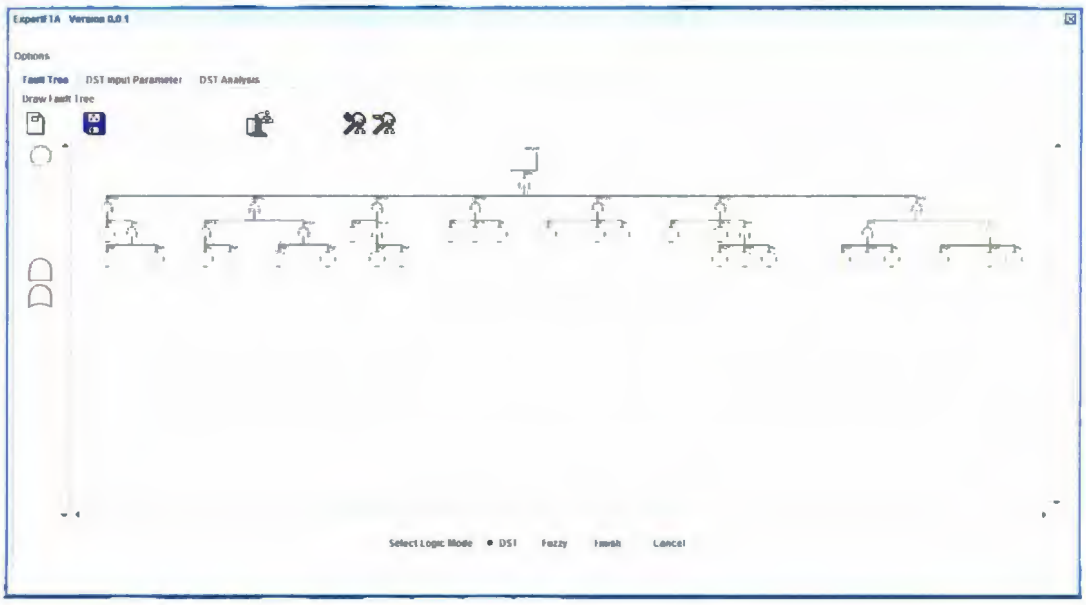


Figure 5.1: Screenshot of the case-study

5.2 Results

Table 5.3 shows the possible combinations of analyses conducted on the fault tree using the same input values. Appendix-A provides the detailed output generated from these studies. Table-5.4 illustrates the result observed from each analysis. Using the fuzzy-approach along with single expert's opinion as input gives identical results that stated in Sadiq *et el.* (2008).

Table 5.3: Possible combinations of analysis

	Single Expert	***Multiple Experts***
Fuzzy-approach	X	X
Evidence theory	X	X

Table 5.4: Probability of water quality failure in Walkerton Ontario

Feature	*Uncertainty Range*	*Point Estimation*
Subjective uncertainty (Fuzzy)	**Single Expert:** [0.0, 0.0019, 0.0104]	**Single Expert:** 0.0019
	Multiple Experts: [0.0001, 0.0019, 0.0114]	**Multiple Experts:** 0.0019
Uncertainty due to ignorance (DST)	**Single Expert:** 0.0237 0.0267	**Bet:** 0.0252
	Multiple Experts: 0.0088 0.0124	**Bet:** 0.0106

Table 5.5 depicts a comparison of error robustness performed using fuzzy approach, evidence theory and traditional approach as per *Ferdous et el* (2010). It is observed that by introducing 20% uncertainty in input data, the fuzzy and evidence theory-based approaches measured almost 0.24% and 9% deviation respectively, whereas 65% deviation is observed in the result of traditional approach. Therefore one can state that the fuzzy based approach is more robust compared to other approaches; moreover introducing multiple experts' opinion definitely gives improved results.

Table 5.5: Error propagation for different approaches

Approaches	Likelihood of VCE (OE_1) *Defuzzified value/ Bet / Deterministic estimation		D (Percentage Deviation)
	Estimated with 20% uncertainty	Estimated with no uncertainty	
Fuzzy-based	0.413	0.412	0.24%
Evidence theory-based	0.328	0.360	8.88%
Traditional	0.126	0.360	65.00%

* Defuzzified estimation for the fuzzy-based approach, the Bet measure for the evidence theory-based approach and deterministic estimation for traditional are used to estimate the likelihood

3 Summary

In this chapter a case study is performed using different analysis approaches. The results of the study along with the discussion on the error robustness of different approaches are also presented.

Chapter 6 COMPARISIONS, CONCLUSION AND RECOMMENDATIONS

This chapter provides a comparative study between ***ExpertFTA*** and available FTA tools. Latter, conclusions and recommendation for the future work is provided.

6.1 Comparative study

In 1970 (Henley, 1981), the first computer based FTA was developed. There are several techniques such as Diagraph Based Methods (Lapp and Powers, 1977 and 1979), fault tree construction by formal methods (Fussell, 1973), Rule Based Methods (Elliott, 1994), and Loop Based Methods (Shafaghi, 1988) which are adopted for FTA-tool. Many researchers and software companies have worked on automated FTA-tools. Brief overview of some of those FTA-tools are given below:

CARA-Fault Tree: CARA-Fault Tree (CARA-Fault Tree, version 4.1, 1999) is a Microsoft Windows based program for top-down construction of fault trees. The modularization technique is implemented here. The complete tree is scanned to check for the repeated input in any sub-trees. Each module is treated as an input event and the probability calculation are done recursively for every module. This tool has implemented the MOCUS (Minimum Obtained Cut-Sets) algorithm to produce minimal cut-sets. Each module will appear as an input event in the minimal cut-sets. So the modularization option should not be used if the user requires the list of minimal cut-sets.

Fault Tree+: Isograph Ltd.® has Developed *Fault Tree+* which runs under the Microsoft platform. This tool (Version 11.0, Demo) can handle large and complex fault tree. It can also analyze and produce full minimal cut-sets.

Windchill FTA: Windchill FTA (formerly Relex Fault Tree) was introduced by PTC®. This tool provides an efficient user interface that allows users to create a fault tree on a window. It uses the idea of modeling a fault-tree in terms of a series of smaller fault trees. An analytical approach is used to calculate and display the probabilities of the events and gates at a mission time. Key features are: enforcing correct fault tree logic, outputting key system metrics which includes unreliability, unavailability, frequency and number of failures etc.

PROFAT: The package PROFAT (**PRO**babilistic **FA**ult **T**ree analysis) (Khan and Abbasi, 1999) is based on an analytical simulation methodology. This package reduces the imprecision and ambiguity of probability analysis of a fault tree by incorporating probability analysis with fuzzy sets. Although this package has the capability to analyze large and complex trees using a modularization technique, it has no graphical interface. User can not draw a fault tree by using this package.

A qualitative comparison amongst a selected set of FTA-tools and ***ExpertFTA*** is performed. From the results shown in Table 6.1 it is observed that the most of the tools such as ***Windchill FTA*** , ***CARA-Fault Tree***, **FaultTree+** and **PROFAT** are unable to handle ignorance in data, knowledge aggregation from multiple-experts and user-define fuzzy linguistic scale. Except **PROFAT**, rest of the tools do not consider vagueness and ambiguity in input data. Though **PROFAT** handles subjective uncertainty by implementing fuzzy set; it fails to account *epistemic uncertainty* owing to ignorance or incompleteness of an expert's knowledge (Ferdous

et al., 2010). And also, this package does not provide a graphical user interface to draw and visualize a fault tree.

On the other hand, the proposed tool, ***ExpertFTA,*** has features which make it unique. ***ExpertFTA*** not only deals with uncertainty but also handles incompleteness and ignorance in the input data. One powerful feature of this tool is that the user can define a suitable linguistic fuzzy-scale such that the analysis is not restricted or stuck with pre-defined fuzzy-values. In each case of analysis (Fuzzy/DST), the user is able to input basic event's value by aggregating knowledge from multiple experts. Moreover, for each basic event, the user can choose the scale type from three options: "Fuzzy Scale", "Most Likely Value" and "Crisp". During fuzzy-analysis, the user can observe the sensitivity by performing alpha-cut calculation. During DST analysis, on the result panel, "Belief", "Plausibility" and "Bet" estimation is provided for each gate. This tool also generates an XML file (in future which will be used to draw a fault tree; in other words, application will automatically draw a fault tree from a given XML file) and a PDF report. Table 6.1 compares features of different FTA-tools.

Table 6.1: Feature comparison of proposed tool with available FTA tools

Feature	[1] *Windchill FTA*	[1]*CARA-Fault Tree*	[1]FaultTree+	PROFAT	[2]ExpertFTA
Subjective (fuzzy-based)	NC	NC	NC	C	C
Ignorance (DST based)	NC	NC	NC	NC	C
knowledge aggregation(multi-Expert)	NC	NC	NC	NC	C
User defined Fuzzy-scale	NC	NC	NC	NC	C
Different scales(Fuzzy, most-likely, crisp)	NC	NC	NC	NC	C
Alpha-cut (fuzzy-based)	NC	NC	NC	NC	C
Bet estimation(DST based)	NC	NC	NC	NC	C
FOD	NC	NC	NC	NC	C

[1]A commercial software,[2] proposed FTA-tool, NC = not considered, C = considered

6.2 Conclusions and Recommendations

FTA is frequently used to evaluate the operational performance, reliability predictions, lifetime, and system safety of complex systems. Developing a software which will handle FTA is quiet sophisticated and complex task. This thesis introduces a software tool- ***ExpertFTA*** to draw and analyze a fault tree using fuzzy and evidence theory approach. Following sub-sections describe the conclusions and suggestions to improve this work.

6.2.1 Conclusions

As already mentioned (chapter 1) that to overcome the limitations in the conventional approach of FTA, Ferdous *et al.* (2009a, 2009b) introduced two different approaches (i.e., fuzzy-based and evidence theory-based approaches). ***ExpertFTA*** is a software tool which incorporated these two approaches efficiently. The following are the key features of ***ExpertFTA***:

- It provides a user a friendly interface to draw and modify large and complex fault tree.
- It allows users to analyze a fault tree using Fuzzy and DST approaches. So that users can compare the results and prepare a valuable decision.
- In the Fuzzy approach, users are able to construct their own fuzzy-scale to input data, which eliminates the restriction of any pre-defined fuzzy-scale.
- It supports three different options for input: Fuzzy Scale, Most-Likely Value and Crisp Value.
- It has the ability to aggregate knowledge from multiple experts for both Fuzzy and DST approaches.
- In DST, users are able to choose from different combination methods (e.g, Yegar, DS) for each basic event input.
- Users can perform sensitivity analysis during the Fuzzy route by alpha-cut evaluation.
- Results from both approaches are displayed as gate-by-gate probability calculation.

- In the DST result-panel, Belief, Plausibility and Bet estimation are displayed for each Gate.
- All inputs in the current session of the application are captured and stored into an XML file for future reference.
- A well-formatted report can be generated.

By observing the above points, and comparing with existing FTA-tools, this tool is more comprehensive and robust in all aspects.

6.2.2 Recommendations

The following recommendations are made to improve this tool:

- ***ExpertFTA*** only deals with independent events; dependency between basic events needs to be considered.
- To get more robust predictions of a system is reliability/failure probability, detailed sensitivity analysis should be introduced (i.e., fuzzy weighted index and cut-sets importance estimations).
- Currently, in the DST analysis mode, there are only two predefined FOD; an attempt should be made to make it user defined and incorporate other recent concepts.
- Although the current version of ***ExpertFTA*** can generate XML of a fault tree, this cannot auto generate a fault tree from a given XML. Handling this issue in the future would provide more usability and flexibility of this tool.

References

Abrahamsson, M. (2002). *Uncertainty in quantitative risk analysis - characterisation and methods of treatment [elektronisk resurs]* Fire Safety Engineering and Systems Safety.

Agarwal, H., Renaud, J. E., Preston, E. L., & Padmanabhan, D. (2004). Uncertainty quantification using evidence theory in multidisciplinary design optimization. *Reliability Engineering & System Safety, 85*(1-3), 281. doi:DOI: 10.1016/j.ress.2004.03.017"

AIChE. (2000). *Guidelines for chemical process quantitative risk analysis* (2nd Edition ed.). New York, USA: Center for Chemical Process Safety/AIChE. Retrieved from http://www.knovel.com/web/portal/browse/display?_EXT_KNOVEL_DISPLAY_bookid=677

Alexander, C. (Ed.). (1979). *The timeless way of building* Oxford University Press.

Alexander, C., Sara, I., & Murray, S. (1977). *A pattern language- town, buildings, constructions* Oxford University Press.

Apostolakis, G. (1990). The concept of probability in safety assessments of technological systems. *Science, 250*(4986), 1359-1364.

Ayyub Bilal M., & Klir George J. (2006). *Uncertainty modeling and analysis in engineering and the sciences*. Boca Raton, FL 33487-2742,US: Chapman & Hall.

Ayyub, B. M. (2001).

A practical guide on conducting expert-opinion elicitation of probabilities and consequences for corps facilities (Technical report No. IWR Report 01-R-01). Alexandria, VA 22315-3868,US: U.S. Army Corps of Engineers. (Expert-Opinion)

Ayyub, B. M. (1991). Systems framework for fuzzy sets in civil engineering. *Fuzzy Sets Syst., 40*(3), 491-508. doi:10.1016/0165-0114(91)90174-O

Ayyub, B. M. (2003). *Risk analysis in engineering and economics.* New York, USA: Chapman and Hall/CRC.

Bae, H., Grandhi, R. V., & Canfield, R. A. (2004). An approximation approach for uncertainty quantification using evidence theory. *Reliability Engineering & System Safety, 86*(3), 215. doi:DOI: 10.1016/j.ress.2004.01.011"

Basili, V. R., & Turner, J. (Dec. 1975). Iterative enhancement: A practical technique for software development

. *IEEE Trans. Software Eng.,* , pp. 390- 396.

Bell, D. (2001). *Software engineering, A programming approach.* (3rd ed.). Reading, MA.: Addison Wesley.

Berztiss, A. T. (2001). Uncertainty management.

Handbook of Software engineering and knowledge engineering () World Scientific.

Billington, R., & Allen, R. N. (1986). *Reliability evaluation of engineering system, marshfield, MA: Pitman publishing inc..*

Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer, 21(5), 61-72.,*

Cheng , Y. (2000). *Uncertainties in fault tree analysis. Tamkang Journal of Science and Engineering, 3*(1), 23-29.

Chojnacki, E., Mercat-Rommens, C., & Baudrit, C. (2005). Influence of mathematical modeling of knowledge. application to the transfer of radionuclides in the environment. *Presented at Workshop on the Evaluation of Uncertainties in Relation to Severe Accidents and Level II Probabilistic Safety Analysis,* Cadarache, France.

Crowl, D. A., & Louvar, J. F. (2001). ***Chemical process safety, fundamentals with applications*** (2nd ed.). Upper Saddle River, New Jersey, USA: Prentice Hall PTR.

Daneshkhah, A. (2004). ***Uncertainty in probabilistic risk assessment: A review***. Unpublished manuscript.

Dezert, J., & Smarandache, F. (2004). Presentation of DSmT. In F. Smarandache, & J. Dezert (Eds.), *Advances and applications of DSmT for information fusion* (pp. 1-32). Rehoboth, USA: American Research Press.

Ferdous, R., Khan, F., Sadiq, R., Amyotte, P., & Veitch, B. (2009). Handling data uncertainties in event tree analysis. *Process Safety and Environmental Protection, 87*(5), 283. doi:DOI: 10.1016/j.psep.2009.07.003"

Ferdous, R., Khan, F., Sadiq, R., Amyotte, P., & Veitch, B. (2010). Fault and event tree analyses for process systems risk analysis: Uncertainty handling formulations. *Risk Anal,* Retrieved from http://www.biomedsearch.com/nih/Fault-Event-Tree-Analyses-Process/20731791.html

Ferdous, R., Khan, F., Sadiq, R., Amyotte, P., & Veitch, B. (2010b). Analyzing system safety and risks under uncertainty using a bow-tie diagram: An innovative approach. *Process Safety and Environment Protection, Submitted*

Ferdous, R., Khan, F., Veitch, B., & Amyotte, P. R. (2009). Methodology for computer aided fuzzy fault tree analysis. *Process Safety and Environmental Protection, 87*(4), 217. doi:DOI: 10.1016/j.psep.2009.04.004"

Ferson, S., Hajagos, J., Berleant, D., Zhang, J., Tucker, W. T., Ginzburg, L., et al. (2004). *Dependence in dempster-shafer theory and probability bounds analysis**. US: Sandia National Laboratories.

Floyd, C. (1984). A systematic look at prototyping, in: Budde, R., kuhlenkamp, K., mathiassen, L. and zullighoven, H. (eds.) approaches to prototyping, In 1. Springer-Verlag: Heidelberg (Ed.), () Springer-Verlag: Heidelberg, 1-17.

Gamma, E. (1995). *Design patterns: Elements of reusable object-oriented software*. Reading, Mass. ; Don Mills, Ont.: Addison-Wesley.

Guth, M. A. S. (1991). A probabilistic foundation for vagueness and imprecision in fault-tree analysis. *Reliability, IEEE Transactions on, 40*(5), 563-571.

Haasl, F. D. (1965). Advanced concepts in fault tree analysis. *System Safety Symposium,* Boeing Company, Seattle, Washington.

Henley, E. J., & Kumamoto, H. (1981). *Reliability engineering and risk assessment* New Jersey: Prentice-Hall Inc.

Hewett, R. Information-based risk assessment software architecture. *IEEE International Engineering Management Conference, v II, p 574-578, 2005, IEMC 2005: 2005 IEEE International Engineering Management Conference, Proceedings,*

HSE. (1996). ***Offshore accident and incident statistics report*** (Technical No. OTO96.954)Health and Safety Executive.

Huang, D., Chen, T., & Wang, M. J. (2001). A fuzzy set approach for event tree analysis. *Fuzzy Sets and Systems, 118*(1), 153-165. doi:DOI: 10.1016/S0165-0114(98)00288-7

Jia, X. (2000). *Object-oriented software development using java: Principles, patterns, and frameworks*. Reading, Mass. ; Don Mills, Ont.: Addison-Wesley.

Khan, F. I., & Abbasi, S. A. (1999). **PROFAT: A user friendly system for probabilistic fault tree analysis**. *Process Safety Progress, 18*(1), 42-49.

Khan, F. I., & Abbasi, S. A. (2001). Risk analysis of a typical chemical industry using ORA procedure. *Journal of Loss Prevention in the Process Industries, 14*(1), 43. DOI: 10.1016/S0950-4230(00)00006-1"

Knight, J. C. (2002). Safety critical system: Challenges and direction *Proceedings - International Conference on Software Engineering, p 547-550,*

Kumamoto, H., & Henley, J. E. (1996). *Probablistic risk assessment and management for engineers and scientists* (2nd ed.) Wiley-IEEE Press.

Lapp, S. A., & Powers, G. J. Computer-aided synthesis of fault-trees. *IEEE trans. reliab.* **1977**, *2*, R-26.

Lees, F. P. (2005). In Mannan S., O'Connor M. K. (Eds.), *Loss prevention in the process industries* (3rd ed.) Elsevier.

Li, H. (2007). *Hierarchical risk assessment of water supply systems.* Unpublished Doctor of Philosophy, Loughborough University, UK,

Lin, C., & Wang, M. J. (1997). Hybrid fault tree analysis using fuzzy sets. *Reliability Engineering & System Safety, 58*(3), 205-213. doi:DOI: 10.1016/S0951-8320(97)00072-0

Liu, Y., Chen, Y., Gao, F., & Jiang, G. (2005). Risk evaluation using evidence reasoning theory. *Machine Learning and Cybernetics, 5*(1), 2855-2860.

Mansfield, D. P., Kletz, T. A., & Al-Hassn, T. (1996). **Optimizing safety by inherent offshore platform design**. *Proceedings of 1996 OMAE Conference on Safety and Reliability, , -Volume II).*

Markowski, A. S., Mannan, M. S., & Bigoszewska, A. (2009). Fuzzy logic for process safety analysis. *Journal of Loss Prevention in the Process Industries, 22*(6), 695-702. DOI: 10.1016/j.jlp.2008.11.011

McCormick, N. J. (1981). *Reliability and risk analysis,* New york: Academic press.

Nan, Feng (School of Management, Tianjin University), Minqiang, L., Jing, X., & Deying, F. (2009). A data-driven model for software development risk analysis using bayesian networks. *IEEE Symposium on Advanced Management of Information for Globalized Enterprises, AMIGE 2008 - Proceedings, p 41-45,*

Patterson-Hine, F. A., & Koen, B. V. (Jun 1989). Direct evaluation of fault trees using object-oriented programming techniques. *IEEE Transactions on Reliability, v 38, n 2, p 186-192,*

Pula, C. R. (2005). *An integrated system for fire and explosion consequence analysis of offshore process facilities* Unpublished M.Eng., Memorial university of Newfoundland, St John's, NL, Canada.

Ren, J., Jenkinson, I., Wang, J., Xu, D. L., & Yang, J. B. (2009). An offshore risk analysis method using fuzzy bayesian network. *Journal of Offshore Mechanics and Arctic Engineering, 131*(4), 041101. doi:10.1115/1.3124123

Ross, J. T. (2004). *Fuzzy logic with engineering applications* (2nd ed.). West Sussex, England.: John Wiley & Sons, Ltd,.

Royce, W. W. (1970). Managing the development of large software systems: Concepts and techniques. *WESCON Technical Papers, Western Electronic show and Convention; v. 14 Reprinted in Proceeding of the Ninth International Conference on Software Engineering, (1989) ACM Press, 328-338.,*

Sadiq, R., Saint-Martin, E., & Kleiner, Y. (2008). Predicting risk of water quality failures in distribution networks under uncertainties using fault-tree analysis. *Urban Water Journal, 4*(5), 287.

Sadiq, R., Kleiner, Y., & Rajani, B. (2007). Water quality failures in distribution Networks—Risk analysis using fuzzy logic and evidential reasoning. *Risk Analysis, 27*(5), 1381-1394.

Sentz, K., & Ferson, S. (2002). *Combination of evidence in dempster-shafer theory* (Technical. US Department of Energy: Sandia National Laboratories.

Sommerville, I. (2001). *Software engineering, (sixth edition)* (6th ed.). Reading, MA: Addison Wesley.

Tanaka, H., Fan, L. T., Lai, F. S., & Toguchi, K. (1983). Fault-tree analysis by fuzzy probability. *IEEE Transactions on Reliability, 32*(5), 453-457.

Thacker, B., & Huyse, L. (2003). Probabilistic assessment on the basis of interval data. *44th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Conference,* Norfolk, Virginia.

Vesely, W. E., Goldberg, F. F., Roberts, N. H., & Haasl, F. D. (1981). *Fault tree handbook.* Washington, DC, USA: U.S. Nuclear Regulatory Commission.

Wagholiar, S. A. (2007). *Acquisition of fuzzy measures in multicriteria decision making using similarity-based reasoning.* Unpublished Doctor of Philosophy, Griffith University, Gold Coast, Australia.

Wang, Y. (2004). Development of a computer-aided fault tree synthesis methodology for quantitative risk analysis in the chemical process industry. (Doctor of Philosophy, Texas A&M University). , 1-164.

Wilcox, R. C., & Ayyub, B. M. (2003). Uncertainty modeling of data and uncertainty propagation for risk studies. *Uncertainty Modeling and Analysis, International Symposium on, 0*, 184. doi:http://doi.ieeecomputersociety.org/10.1109/ISUMA.2003.1236160

Wu, H. (2006). Fuzzy bayesian system reliability assessment based on exponential distribution. *Applied Mathematical Modelling, 30*(6), 509-530. doi:DOI: 10.1016/j.apm.2005.05.014

Yager, R. R. (1987). On the dempster-shafer framework and new combination rules. *Information Sciences, 41*(2), 93-137. doi:DOI: 10.1016/0020-0255(87)90007-7

Yang, Z., Suzuki, K., Shimada, Y., & Sayama, H. (1995). **Fuzzy fault diagnostic system based on fault tree analysis**. *Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and the Second International Fuzzy Engineering Symposium., Proceedings of 1995 IEEE International Conference on, 1*, 165-170.

Yuhua, D., & Datao, Y. (2005). Estimation of failure probability of oil and gas transmission pipelines by fuzzy fault tree analysis. *Journal of Loss Prevention in the Process Industries, 18*(2), 83-88. doi:DOI: 10.1016/j.jlp.2004.12.003

Zadeh, L. (1965). Fuzzy sets. *Fuzzy Sets. Information and Control, 8*, 338-353.

Appendix – A

1. Analysis report from Single Expert using Fuzzy:

Analysis Report
Creation Time: Tue Jan 31 20:55:12 NST 2012
Logic Mode: Fuzzy

Gate's Value

Gate	P1	P2	P3
GATE_1	0.0000	0.0019	0.0104
GATE_2	0.0750	0.1441	0.2450
GATE_3	0.2500	0.3600	0.4900
GATE_4	0.0000	0.0133	0.0231
GATE_5	0 6399	0 4410	0 2879
GATE_6	0 9098	0 7199	0 5600
GATE_7	0 0000	0 1001	0 2001
GATE_8	0.0902	0 2001	0.3000
GATE_9	0.0000	0.0631	0.0961
GATE_10	0.7999	0.6300	0.4799
GATE_11	1.0000	0.9919	0.9642
GATE_12	0.0000	0.0041	0.0181
GATE_13	0 0000	0 0041	0 0181
GATE_14	0.0000	0 0000	0.0011
GATE_15	0 0000	0 0041	0 0181

Basic Event's Value

Basic Event ID	P1	P2	P3
BE_1	0.3000	0.4001	0.5000
BE_2	0.5000	0.6000	0.7000
BE_3	0.5000	0.6000	0.7000
BE_4	0.2001	0.3000	0.4001
BE_5	0.2001	0.3000	0.4001
BE_6	0.0000	0.1001	0.2001
BE_7	0.0000	0.1001	0.2001
BE_8	0.2001	0.3000	0.4001
BE_9	0.9001	1.0000	1.0000
BE_10	0.0000	0.1001	0.2001
BE_11	0.9001	1.0000	1.0000
BE_12	0.1001	0.2001	0.3000
BE_13	0.0000	0.1001	0.2001
BE_14	0.0000	0.1001	0.2001
BE_15	0.2001	0.3000	0.4001
BE_16	0.0000	0.1001	0.2001
BE_17	0.1001	0.2001	0.3000
BE_18	0.1001	0.2001	0.3000
BE_19	0.0000	0.1001	0.2001
BE_20	0.1001	0.2001	0.3000
BE_21	0.1001	0.2001	0.3000
BE_22	0.0000	0.0000	0.1001
BE_23	0.0000	0.0000	0.1001
BE_24	0.0000	0.0000	0.1001
BE_25	0.0000	0.1001	0.2001
BE_26	0.1001	0.2001	0.3000
BE_27	0.1001	0.2001	0.3000

Expert's Opinion

Be ID	Expert No	*P1*	*P2*	*P3*
BE_1	Expert_1	0.3	0.4	0.5
BE_2	Expert_1	0.5	0.6	0.7
BE_3	Expert_1	0.5	0.6	0.7
BE_4	Expert_1	0.2	0.3	0.4
BE_5	Expert_1	0.2	0.3	0.4
BE_6	Expert_1	0.0	0.1	0.2
BE_7	Expert_1	0.0	0.1	0.2
BE_8	Expert_1	0.2	0.3	0.4
BE_9	Expert_1	0.9	1.0	1.0
BE_10	Expert_1	0.0	0.1	0.2
BE_11	Expert_1	0.9	1.0	1.0
BE_12	Expert_1	0.1	0.2	0.3
BE_13	Expert_1	0.0	0.1	0.2
BE_14	Expert_1	0.0	0.1	0.2
BE_15	Expert_1	0.2	0.3	0.4

BE_16				
	Expert_1	0 0	0 1	0 2
BE_17				
	Expert_1	0 1	0 2	0 3
BE_18				
	Expert_1	0 1	0 2	0 3
BE_19				
	Expert_1	0.0	0.1	0.2
BE_20				
	Expert_1	0.1	0.2	0.3
BE_21				
	Expert_1	0 1	0 2	0 3
BE_22				
	Expert_1	0 0	0 0	0 1
BE_23				
	Expert_1	0 0	0 0	0.1
BE_24				
	Expert_1	0.0	0.0	0.1
BE_25				
	Expert_1	0.0	0.1	0 2
BE_26				
	Expert_1	0.1	0 2	0 3
BE_27				
	Expert_1	0 1	0 2	0 3

2. Analysis report from Multiple Experts using Fuzzy:

Analysis Report
Creation Time: Mon Feb 06 21:02:38 NST 2012
Logic Mode: Fuzzy

Gate's Value

Gate	*P1*	*P2*	*P3*
GATE_1	0.0001	0.0019	0.0114
GATE_2	0.0151	0.0480	0.1051
GATE_3	0.0501	0.1200	0.2101
GATE_4	0.0300	0.0511	0.0615
GATE_5	0.7476	0.5670	0.3840
GATE_6	0.8953	0.7460	0.5727
GATE_7	0.0268	0.0902	0.1934
GATE_8	0.0801	0.1801	0.2901
GATE_9	0.0259	0.0631	0.0960
GATE_10	0.7733	0.6300	0.4800
GATE_11	0.9993	0.9919	0.9644
GATE_12	0.0004	0.0041	0.0180
GATE_13	0.0004	0.0041	0.0180
GATE_14	0.0001	0.0011	0.0081
GATE_15	0.0004	0.0041	0.0180

Basic Event's Value

Basic Event ID	*P1*	*P2*	*P3*
BE_1	0.3000	0.4000	0.5000
BE_2	0.5000	0.6000	0.7001
BE_3	0.1001	0 2000	0 3000
BE_4	0.2001	0.3000	0 4000
BE_5	0 2000	0 3000	0 4000
BE_6	0 0334	0 1001	0 2000
BE_7	0 0334	0 1001	0 2001
BE_8	0 2000	0 3000	0 4000
BE_9	0.8001	0 9001	0 9667
BE_10	0.0334	0 1001	0 2000
BE_11	0.8001	0.9001	0.9667
BE_12	0.1001	0.2000	0.3000
BE_13	0.0334	0 1001	0.2000
BE_14	0.0334	0 1001	0.2000
BE_15	0.2000	0 3000	0 4000
BE_16	0 0334	0 1001	0 2000
BE_17	0 1001	0 2000	0 3000
BE_18	0 1001	0 2000	0 3000
BE_19	0.0334	0 1001	0 2000
BE_20	0 1001	0 2000	0 3000
BE_21	0.1001	0.2000	0.3000
BE_22	0.0334	0.1001	0.2001
BE_23	0.0334	0.1001	0.2001
BE_24	0.0334	0.1001	0.2001
BE_25	0.0334	0.1001	0.2000
BE_26	0.1001	0 2000	0.3000
BE_27	0.1001	0.2000	0.3000

Expert's Opinion

Be ID	Expert No	*P1*	*P2*	*P3*
BE_1				
	Expert_1	0.3	0.4	0.5
	Expert_2	0.4	0.5	0.6
	Expert_3	0.2	0.3	0.4
BE_2				
	Expert_1	0.5	0.6	0.7
	Expert_2	0.4	0.5	0.6
	Expert_3	0.6	0.7	0.8
BE_3				
	Expert_1	0.1	0.2	0.3
	Expert_2	0.0	0.1	0.2
	Expert_3	0.2	0.3	0.4
BE_4				
	Expert_1	0.2	0.3	0.4
	Expert_2	0.3	0.4	0.5
	Expert_3	0.1	0.2	0.3
BE_5				
	Expert_1	0.2	0.3	0.4
	Expert_2	0.1	0.2	0.3
	Expert_3	0.3	0.4	0.5
BE_6				
	Expert_1	0.0	0.1	0.2
	Expert_2	0.0	0.0	0.1
	Expert_3	0.1	0.2	0.3
BE_7				
	Expert_1	0.0	0.1	0.2
	Expert_2	0.1	0.2	0.3
	Expert_3	0.0	0.0	0.1

BE_8				
	Expert_1	0.2	0.3	0.4
	Expert_2	0.1	0.2	0.3
	Expert_3	0.3	0.4	0.5
BE_9				
	Expert_1	0.9	1.0	1.0
	Expert_2	0.8	0.9	1.0
	Expert_3	0.7	0.8	0.9
BE_10				
	Expert_1	0.0	0.1	0.2
	Expert_2	0.0	0.0	0.1
	Expert_3	0.1	0.2	0.3
BE_11				
	Expert_1	0.9	1.0	1.0
	Expert_2	0.8	0.9	1.0
	Expert_3	0.7	0.8	0.9
BE_12				
	Expert_1	0.1	0.2	0.3
	Expert_2	0.0	0.1	0.2
	Expert_3	0.2	0.3	0.4
BE_13				
	Expert_1	0.0	0.1	0.2
	Expert_2	0.0	0.0	0.1
	Expert_3	0.1	0.2	0.3
BE_14				
	Expert_1	0.0	0.1	0.2
	Expert_2	0.0	0.0	0.1
	Expert_3	0.1	0.2	0.3

BE_15				
	Expert_1	0.2	0.3	0.4
	Expert_2	0.1	0.2	0.3
	Expert_3	0.3	0.4	0.5
BE_16				
	Expert_1	0.0	0.1	0.2
	Expert_2	0.0	0.0	0.1
	Expert_3	0.1	0.2	0.3
BE_17				
	Expert_1	0.1	0.2	0.3
	Expert_2	0.0	0.1	0.2
	Expert_3	0.2	0.3	0.4
BE_18				
	Expert_1	0.1	0.2	0.3
	Expert_2	0.0	0.1	0.2
	Expert_3	0.2	0.3	0.4
BE_19				
	Expert_1	0.0	0.1	0.2
	Expert_2	0.0	0.0	0.1
	Expert_3	0.1	0.2	0.3
BE_20				
	Expert_1	0.1	0.2	0.3
	Expert_2	0.0	0.1	0.2
	Expert_3	0.2	0.3	0.4
BE_21				
	Expert_1	0.1	0.2	0.3
	Expert_2	0.0	0.1	0.2
	Expert_3	0.2	0.3	0.4
BE_22				
	Expert_1	0.0	0.0	0.1
	Expert_2	0.0	0.1	0.2
	Expert_3	0.1	0.2	0.3

BE_23				
	Expert_1	0.0	0.0	0.1
	Expert_2	0.0	0.1	0.2
	Expert_3	0.1	0.2	0.3
BE_24				
	Expert_1	0.0	0.0	0.1
	Expert_2	0.0	0.1	0.2
	Expert_3	0.1	0.2	0.3
BE_25				
	Expert_1	0.0	0.1	0.2
	Expert_2	0.0	0.0	0.1
	Expert_3	0.1	0.2	0.3
BE_26				
	Expert_1	0.1	0.2	0.3
	Expert_2	0.0	0.1	0.2
	Expert_3	0.2	0.3	0.4
BE_27				
	Expert_1	0.1	0.2	0.3
	Expert_2	0.0	0.1	0.2
	Expert_3	0.2	0.3	0.4

Alpha-Cut Results

Alpha-Cut	*P1*	*P2*	*P3*
0.2	0.0005	0.0019	0.0095
0.35	0.0008	0.0019	0.0081
0.56	0.0012	0.0019	0.0061
0.75	0.0015	0.0019	0.0043
0.15	0.0004	0.0019	0.0100
0.45	0.0010	0.0019	0.0072
0.65	0.0013	0.0019	0.0053

3. Analysis report from Single Expert using DST:

Analysis Report
Creation Time: Sun Feb 19 20:04:29 NST 2012
Logic Mode: DST

Gate's Value

Gate	*Belief*	*Plausibility*	*Bet*
GATE_1	0.0237	0.0267	0.0252
GATE_2	0.4684	0.4687	0.4686
GATE_3	0.5203	0.5205	0.5204
GATE_4	0.3599	0.3449	0.3524
GATE_5	0.4001	0.4002	0.4002
GATE_6	0.4001	0.4251	0.4126
GATE_7	0.0490	0.0490	0.0490
GATE_8	0.1224	0.1223	0.1224
GATE_9	0.2520	0.2928	0.2724
GATE_10	0.0221	0.0221	0.0221
GATE_11	0.0015	0.0015	0.0015
GATE_12	0.0700	0.0699	0.0700
GATE_13	0.9004	0.8841	0.8923
GATE_14	0.0023	0.0023	0.0023
GATE_15	0.0976	0.1139	0.1058

Basic Event's Value

Basic Event	Belief	Plausibility
BE_1	0 9002	0 9003
BE_2	0 8002	0.8003
BE_3	0 6502	0.6503
BE_4	0.5000	0.5000
BE_5	0.8002	0.8003
BE_6	0.8002	0.8502
BE_7	0.5000	0.5000
BE_8	0 4002	0.4003
BE_9	0 6502	0 6503
BE_10	0 6502	0 6503
BE_11	0 7000	0 7501
BE_12	0.6000	0.6000
BE_13	0.6000	0.6503
BE_14	0.1001	0.1001
BE_15	0.4002	0.4003
BE_16	0.5502	0.5503
BE_17	0 2002	0.2002
BE_18	0 1001	0 1001
BE_19	0 5000	0 5000
BE_20	0 3000	0 3000
BE_21	0.8002	0.8003
BE_22	0.3000	0.3000
BE_23	0.0501	0.0501
BE_24	0.1500	0.1500
BE_25	0.6502	0.6503
BE_26	0 3000	0 3502
BE_27	0.5000	0.5000

Expert's Opinion

Be ID	Expert No	*Success*	*Failure*
BE_1	Expert_1	0 1	0 9
BE_2	Expert_1	0 2	0 8
BE_3	Expert_1	0 35	0.65
BE_4	Expert_1	0 5	0 5
BE_5	Expert_1	0 2	0 8
BE_6	Expert_1	0 15	0 8
BE_7	Expert_1	0.5	0 5
BE_8	Expert_1	0.6	0.4
BE_9	Expert_1	0.35	0.65
BE_10	Expert_1	0.35	0.65
BE_11	Expert_1	0.25	0 7
BE_12	Expert_1	0.4	0 6
BE_13	Expert_1	0.35	0 6
BE_14	Expert_1	0.9	0 1
BE_15	Expert_1	0 6	0 4

BE_16			
	Expert_1	0.45	0.55
BE_17			
	Expert_1	0.8	0.2
BE_18			
	Expert_1	0.9	0.1
BE_19			
	Expert_1	0.5	0.5
BE_20			
	Expert_1	0.7	0.3
BE_21			
	Expert_1	0.2	0.8
BE_22			
	Expert_1	0.7	0.3
BE_23			
	Expert_1	0.95	0.05
BE_24			
	Expert_1	0.85	0.15
BE_25			
	Expert_1	0.35	0.65
BE_26			
	Expert_1	0.65	0.3
BE_27			
	Expert_1	0.5	0.5

4. Analysis report from Multiple Experts using DST:

Analysis Report
Creation Time: Sun Feb 05 22:49:33 NST 2012
Logic Mode: DST

Gate's Value

Gate	Belief	Plausibility	Bet
GATE_1	0.0088	0.0124	0.0106
GATE_2	0.6284	0.7851	0.7068
GATE_3	0 6455	0.8009	0.7232
GATE_4	0 1265	0 0512	0 0889
GATE_5	0 1924	0 5204	0 3564
GATE_6	0 8434	0 8934	0.8684
GATE_7	0.1080	0.0606	0.0843
GATE_8	0 2059	0.0807	0.1433
GATE_9	0.6606	0.7877	0.7242
GATE_10	0.0073	0.2157	0.1115
GATE_11	0.0062	0.0136	0.0099
GATE_12	0.3165	0 0474	0.1820
GATE_13	0 9098	0 6099	0 7599
GATE_14	0 0013	0 0438	0 0226
GATE_15	0 0891	0 3622	0.2257

Basic Event's Value

Basic Event	*Belief*	*Plausibility*
BE_1	0.9735	0.9802
BE_2	0.9604	0.9701
BE_3	0.6721	0.8255
BE_4	0.6000	0.8002
BE_5	0.3206	0.6503
BE_6	0.9093	0.9403
BE_7	0.9275	0.9501
BE_8	0.5242	0.7502
BE_9	0.2201	0.4234
BE_10	0.7361	0.8602
BE_11	0.8539	0.9103
BE_12	0.8573	0.9202
BE_13	0.9023	0.9403
BE_14	0.1002	0.5503
BE_15	0.1432	0.5203
BE_16	0.5004	0.7528
BE_17	0.1432	0.5203
BE_18	0.1349	0.5503
BE_19	0.4003	0.7003
BE_20	0.1201	0.2090
BE_21	0.4003	0.8005
BE_22	0.0600	0.0968
BE_23	0.0607	0.5728
BE_24	0.3463	0.7880
BE_25	0.6038	0.8081
BE_26	0.3097	0.6180
BE_27	0.4764	0.7250

Expert's Opinion

Be ID	Expert No	*Success*	*Failure*
BE_1			
	Expert_1	0 1	0.9
	Expert_2	0.2	0 8
	Expert_3	0.3	0 7
BE_2			
	Expert_1	0.2	0.8
	Expert_2	0 1	0 85
BE_3			
	Expert_1	0.35	0 65
	Expert_2	0 45	0 5
BE_4			
	Expert_1	0.5	0.5
	Expert_2	0 4	0 6
BE_5			
	Expert_1	0 45	0 5
	Expert_2	0 7	0 3
BE_6			
	Expert_1	0 2	0 8
	Expert_2	0.25	0.7
BE_7			
	Expert_1	0 15	0 8
	Expert_2	0.25	0.75
BE_8			
	Expert_1	0.5	0.5
	Expert_2	0.45	0.5
BE_9			
	Expert_1	0.6	0.4
	Expert_2	0.45	0 5

BE_10			
	Expert_1	0.35	0.65
	Expert_2	0 4	0 6
BE_11			
	Expert_1	0.25	0.7
	Expert_2	0 3	0 7
BE_12			
	Expert_1	0.4	0.6
	Expert_2	0 2	0 8
BE_13			
	Expert_1	0.35	0.6
	Expert_2	0.15	0 85
BE_14			
	Expert_1	0.9	0.1
	Expert_2	0 5	0 5
	Expert_3	0 7	0 3
BE_15			
	Expert_1	0 6	0 4
	Expert_2	0 8	0 2
BE_16			
	Expert_1	0 45	0.55
	Expert_2	0.55	0.45
BE_17			
	Expert_1	0 8	0 2
	Expert_2	0 6	0 4
BE_18			
	Expert_1	0 9	0 1
	Expert_2	0 3	0 5

BE_19			
	Expert_1	0.5	0.5
	Expert_2	0.6	0.4
	Expert_3	0.6	0.4
BE_20			
	Expert_1	0.7	0.3
	Expert_2	0.6	0.35
BE_21			
	Expert_1	0.2	0.8
	Expert_2	0.5	0.5
BE_22			
	Expert_1	0.7	0.3
	Expert_2	0.8	0.2
BE_23			
	Expert_1	0.95	0.05
	Expert_2	0.45	0.55
BE_24			
	Expert_1	0.85	0.15
	Expert_2	0.25	0.75
BE_25			
	Expert_1	0.35	0.65
	Expert_2	0.55	0.45
BE_26			
	Expert_1	0.65	0.3
	Expert_2	0.5	0.45
BE_27			
	Expert_1	0.5	0.5
	Expert_2	0.5	0.45