PETRI NET MODELING OF FAULT ANALYSIS FOR PROBABILISTIC RISK ASSESSMENT

By

Andrew Lee

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Master of Applied Science

in

The Faculty of Energy Systems and Nuclear Science

University of Ontario Institute of Technology

April, 2013

ABSTRACT

Fault trees and event trees have been widely accepted as the modeling strategy to perform Probabilistic Risk Assessment (PRA). However, there are several limitations associated with fault tree/event tree modeling. These include 1. It only considers binary events; 2. It assumes independence among basic events; and 3. It does not consider timing sequence of basic events. This thesis investigates Petri net modeling as a potential alternative for PRA modeling. Petri nets have mainly been used as a simulation tool for queuing and network systems. However, it has been suggested that they could also model failure scenarios, and thus could be a potential modeling strategy for PRA. In this thesis, the transformations required to model logic gates in a fault tree by Petri nets are explored. The gap between fault tree analysis and Petri net analysis is bridged through gate equivalency analysis. Methods for qualitative and quantitative analysis for Petri nets are presented. Techniques are developed and implemented to revise and tailor traditional Petri net modeling for system failure analysis. The airlock system and the maintenance cooling system of a CANada Deuterium Uranium (CANDU) reactor are used as case studies to demonstrate Petri nets ability to model system failure and provide a structured approach for qualitative and quantitative analysis. The minimal cutsets and the probability of the airlock system failing to maintain the pressure boundary are obtained. Furthermore, the case study is extended to non-coherent system analysis due to system maintenance.

Keywords: Petri net, fault tree, coherent, non-coherent, CANDU, nuclear power plant

ACKNOWLEDGMENTS

I would like to acknowledge all those who stood by me during this journey; professors at UOIT, family, and friends.

Firstly I would like to acknowledge my supervisor, Dr. Lixuan Lu for her guidance, insight, and support which kept me motivated and on track.

I want to thank my family and friends for their never-ending support and patience. Through all the ups and downs, they remained my source of inspiration.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AEC	Atomic Energy Commission
CANDU	CANada Deuterium Uranium
CDF	Core Damage Frequency
DBA	Design Basis Accidents
EQ	Environmental Qualification
FMEA	Failure Mode and Effects Analysis
FRACAS	Failure Reporting, Analysis, and Corrective Action System
FTA	Fault Tree Analysis
FE	Flooding Events
FHSF	Fuel Handling System Failures
FV	Fussell-Vesely
I&C	Instrument and Control
IREP	Interim Reliability Evaluation Program
LERF	Large Early Release of Radioactivity Frequency
LOCA	Loss of Coolant Accidents
LOCECI	Loss of Emergency Coolant Injection
NPC	Negative Pressure Containment
NRC	Nuclear Regulatory Commission

PHT	Primary Heat Transport
PRA	Probabilistic Risk Assessment
QDC	Quick Disconnect Couplings
RSS	Reactor Safety Study
RSSMAP	Reactor Safety Study Methodology Application Program
RBD	Reliability Block Diagram
SSLB	Secondary Side Line Break
SDS1	Shutdown System 1
SPN	Stochastic Petri Net
TMI	Three Mile Island
TPN	Time Petri Net

CHAPTER 1 INTRODUCTION

1.1 Historical Background

Nuclear power plants are highly regulated from design, construction, operation, and decommissioning. Engineers must comply with stringent regulations to limit the risk of radioactivity released to the public. The objectives of nuclear safety are to operate in a safe manner, protect the public and environment, limit the release of radioactive material in the event of a design basis accident, and adhere to the rules and regulations.

A deterministic approach attempts to ensure that the plausible accident scenarios are taken into account, and that the monitoring systems and safety systems will be capable to contain the accident and prevent the release of radioactive material. This approach is based on two principles referred to as the leak tight barrier and the concept of Defense-in-Depth [USNRC, 1980].

Leak tight barriers consist of fuel cladding; the primary reactor coolant system; and the containment building between the radioactive source and the public.

Defense-in-Depth assumes accidents may still occur from equipment failures and human factors despite engineering design safety, and therefore systems are designed and installed to limit the consequences for both the public and environment. Defense-in-Depth incorporates several stages including: prevention and surveillance, protection, and safeguard.

When the Three Mile Island accident occurred in 1979, it became apparent that the deterministic approach had its limitations and how safety assessment techniques should be required. The recommendations made after was to incorporate probabilistic analysis techniques in conjunction with conventional deterministic approaches. The Rasmussen report was the first of its kind to assess the potential risk of core damage for nuclear power reactors [WASH-1400 1975].

The accident at Chernobyl in 1986 revealed the potential consequences of failure to manage nuclear power plant safety. This revealed the urgent need to develop a sustainable probabilistic safety and risk program for accident prevention. Probabilistic Risk Assessment (PRA) and safety assessments have seen much attention and development since, and are now an integral part of nuclear power plant operation and design [Kessides, 2012].

PRA safety assessments were developed to calculate the probability of failure events for complex nuclear and aerospace engineering systems. Modeling techniques such as fault tree analysis and event tree analysis have been widely used to develop scenarios for simulating hypothetical accidents, and estimate the frequency of failure or accidents.

1.2 Problem Statement

The modeling technique to perform PRA should consist of the abilities to perform both qualitative and quantitative analysis, allow dependence among basic events, simulate various scenarios, model coherent and non-coherent systems, and a graphical user interface.

Fault trees and event trees have been widely accepted as the modeling technique to perform PRA. However, there are several limitations associated with the traditional modeling methods. These include

1. It only considers binary events;
2. It assumes independence among basic events;
3. It does not consider timing sequence of basic events; and
4. No graphical user interface for simulation.

Ignoring these limitations constrains the accuracy of the PRA and results in analysis that over-simplifies scenarios and models [Marhavila et al, 2011].

1.3 Objective of the Thesis

Petri nets are most commonly used for work flow management such as: manufacturing, control systems, and queuing applications. Petri nets have two analysis methods: the reachability method and the algebraic method. These methods are used to determine properties of discrete event systems such as: reachability, liveliness, safeness, boundedness and stability, conservation, and controllability. These system properties are essential for system analysis for areas such as manufacturing, real-time processing, computer architecture, dynamic control, supervisory control, and material handling [Peterson, 1981,]. There has been limited work surrounding its ability to provide a structured approach to determine the probability of failure in a complex system. Petri nets have mainly been used as a simulation tool for queuing and network systems. However, it has been suggested that they could also model failure scenarios, and thus be a potential modeling strategy for PRA [Liu, 1997].

Currently, the most widely accepted modeling technique used in the nuclear industry to determine probability of failure in a complex system is the fault tree analysis and event tree analysis. They are used to identify and evaluate the sequence of events in a potential accident scenario, following the occurrence of an initiating event. The objective of the thesis is to introduce Petri net as a potential alternative analysis technique for PRA.

Objective 1, investigate Petri net modeling as a potential alternative modeling strategy for PRA by determining the gaps between Petri net modeling and traditional fault tree modeling, and bridging the gaps. Objective 2, construct a Petri net model of the CANDU Airlock system, and apply qualitative and quantitative methodologies, and verify their results. Objective 3, software development to address limitations with current traditional Petri net software packages for system failure modeling.

1.4 Organization of the Thesis

This thesis comprises of five chapters. The present chapter introduces objectives and contributions of the thesis.

The second chapter presents the literature review which provides the fundamental definitions related to the modeling of a system by Petri nets. Petri net analysis techniques are introduced, and their advantages of limitations are discussed. A brief survey of safety programs; specifically PRA and the current modeling techniques (mainly fault tree and event tree), are also provided.

Chapter three describes the methodologies for Petri net techniques as an alternative modeling tool for PRA. The behaviours and interactions between system components, and construction techniques for a structured model to perform qualitative and quantitative analysis are explored. Petri nets and analysis software are introduced and discussed.

In chapter four, application of Petri net for PRA modeling and analysis of systems in a CANDU reactor is demonstrated. Two cases are studied. In case study 1, the airlock system includes interfaces with mechanical, electrical, and operation factors, and the system is coherent.

Case study 2 explores a non-coherent system analysis when the primary heat transport system is undergoing maintenance and the maintenance cooling system is active.

Conclusions are given in the last chapter. The scope of Petri nets for PRA is discussed. This chapter also includes recommendations for future research.

1.5 Major Contributions

This thesis demonstrates Petri nets as a viable modeling tool for Probabilistic Risk Assessment. This is established through the modeling and analysis techniques for fault models through utilization of logic functions which provide a structured approach to analysis. Petri net models are developed for PRA analysis and simulation. Select CANDU systems are used to demonstrate the techniques introduced in the thesis, with results verified by fault tree analysis. Also, software development is presented for improved Petri net simulation for system failure analysis.

CHAPTER 2 LITERATURE REVIEW

The initial consideration of safety issues began with the Manhattan Project during World War II. During the construction process, there were disputes over safety issues which led to engineers dividing the reactor design into smaller, relatively independent sub-systems, thus allowing any dependent systems to be designed. This is the concept of functional and structural independence, which we know today as Defense-in-Depth. The concept of Defense-in-Depth originated in the 1940's, and was dominated by the lack of precise knowledge of design margins which evolved into a set of regulatory design and safety principles: [Rhodes, 1986]

1. Use of multiple active and/or passive engineered barriers to rule out any single failures leading to the release of radioactive materials.
2. Incorporation of large design margins to overcome any lack of precise knowledge (epistemic uncertainty) about capacity of barriers and magnitude of challenges imposed by normal or accident conditions.
3. Application of quality assurance in design and manufacturing.
4. Operation within predetermined safe design limits.
5. Continuous testing, inspections, and maintenance to preserve original design margins.

   To measure the effectiveness and performance of the safety systems, deterministic approaches were used through conservative assumptions and calculations. The concept of "design basis accidents" was developed to measure the effectiveness of the barriers and safety systems. Based on Design Basis Accidents (DBA), safety was defined as the ability of a nuclear reactor to withstand a fixed set of prescribed accident scenarios which

would be the most significant adverse events in a nuclear power plant. The premise was that if a plant could handle the DBA, it could handle any other accidents. As part of Defense-in-Depth, multiple back-up equipment and redundancies in safety design is required.

In 1956, the first comprehensive study of the consequences of a large nuclear accident, WASH-740 was published by the AEC. The purpose of this document was to aid congress to deliberate on the Price-Anderson Act regarding the potential harms of reactor accidents, and to provide government insurance for private nuclear reactors. WASH-740 estimated the risk for a serious reactor accident as 10^{-6} per reactor's year of operation, which is still used for estimations of large, early releases of radiation due to reactor accidents today. The study focused on the dangers of large Loss of Coolant Accidents (LOCAs) as the leading source of radiation released into the environment. In 1967, AEC realized that under some circumstances, the containment building may fail. The key to protect the health and safety of the public was shifted to prevent accidents severe enough to threaten containment. In 1974, the Nuclear Regulatory Commission (NRC) was created to replace the AEC due to conflicts of interest to promote and regulate the nuclear program [Ford, 1977]. The NRC is responsible for civilian nuclear power regulation and assuring the protection of the health and safety of the public.

In 1972, Senator John O. Pastore helped initiate the Reactor Safety Study (RSS), also known as the Rasmussen Report (WASH-1400), with the purpose of extending the Price-Anderson Act, and also to ease public concerns over the nuclear technology and licensing which were seen as inadequate and inconsistent with respect to the apparent safety significance of various systems, structures, and components within the plant. Fault trees were the main tools used to develop almost all of the major safety related systems but it was realized that integrating

fault tree analysis for the entire plant was too complex, given time and resource constraints. This led to the development of the event tree concept to model the time-line of the possible accidents scenarios.

The WASH-1400 analyzed six specific LOCAs:

1. Large pipe breaks (3” to 6” in diameter)
2. Small/intermediate pipe breaks (2” to 6” in diameter)
3. Small pipe breaks (less than 2” in diameter)
4. Large disruptive reactor vessel ruptures
5. Gross steam generator ruptures
6. Ruptures in systems that interface with the reactor coolant system

Besides LOCAs, the RSS investigated several types of reactor transients as possible initiating events for reactor system failures. The RSS defined transient as any significant deviation from the normal operating value of any of the key reactor operating parameters including all non-LOCA situations that could lead to fuel heat imbalances. Transients could occur from a variety of means, such as equipment failure or human error. Transients were divided into two categories, anticipated, such as loss of off-site power, and unanticipated, such as reactor vessel rupture.

Following the modeling and analysis of the reactor during an accident, an analysis was completed for the potential radiation release from the reactor into the containment and into the environment. Once the radiation release was known, the consequences of expected human, economic, and environmental loss could be estimated. The RSS showed that most accidents that led to radiation release would only have small consequences.

2.1 Probabilistic Risk Assessment

In 1975, the WASH-1400 [NUREG-75/014, 1975] was published on the Surry and Peach Bottom nuclear power plants. After publication of WASH-1400, the probabilities and consequences of nuclear plant accidents had been adequately addressed and that, as a technology, those risks were very small. However, the critical review by the Lewis Committee in 1978 [NUREG/CR-0400, 1978], and by others [NRC, 1979] in 1979, raised serious questions about the uncertainties surrounding the numerical assessment of risk, which prompted the commission to issue its somewhat cautious policy statement on the WASH-1400 report and on the future use of PRA in the regulation of nuclear power. One of the concerns was the use of lognormal distribution to model the probability of failure uncertainties [NUREG/CR-0400, 1978]. Though the committee indicated this problem, they did not have a better solution on how to account for the 10-1000 fold uncertainties in failure probabilities due to limited data. The comments by the Lewis Committee and others was that despite the risks being small, the uncertainties were large and pervasive – mostly due to questions regarding proper modeling of the accident sequences, lack of understanding of the phenomenology of the progression of accidents involving severe core degradation or core melt, completeness, and the lack of an adequate database for very low probability but high consequence accidents. The committee concluded that the RSS accident probabilities were not considered reliable for the overall risk of reactor accidents [US NRC, 1978].

Following the Lewis Committee Report of 1978, the NRC withdrew its support of the RSS results and disavowed the Executive Summary. Around this time, the accident at Three Mile Island (TMI) Unit 2 occurred in March 1979. This event indicated that major accidents not addressed in the formal reactor licensing process were possible, and that new approaches to

nuclear regulation were required [US NRC, 1978]. The RSS had considered a similar sequence of events and showed that this sequence was not among the risk-significant contributors for that reactor design. The TMI accident confirmed a major RSS insight that small LOCAs are more risk-significant than the large LOCAs, which the NRC used as a design basis accident for worst-case LOCAs in licensing reactors. The RSS also pointed out the potential role of human error, which was a huge significance in the TMI accident when operators turned off the ECCS. As a result, the NRC restored all papers and documents to reference the RSS.

After the TMI accident, the NRC began to devote additional resources towards the expansion of PRA use in the industry. During 1979-1982, the NRC undertook two sets of follow-up PRA studies. The Reactor Safety Study Methodology Application Program (RSSMAP) was to apply the RSS methodology to additional reactor designs [NUREG/CR-1659, 1981], and the Interim Reliability Evaluation Program (IREP) [NUREG/CR-1659, 1981], was a planned multi-plant reliability evaluation program to develop and standardize the reliability methodology involved in performing reliability and safety studies. In 1983, the NUREG/CR-2300 PRA procedures guide was published. In 1990, the NRC provided additional guidance regarding frequency of core damage accidents and Large Early Release of Radioactivity Frequency (LERF) [NUREG/CR-1659, 1981]. The numerical value of Core Damage Frequency (CDF) is one in ten thousand and LERF is one in one hundred thousand. These values later evolved into the benchmark values of 10^{-4} for CDF and 10^{-5} for LERF. By 1995, the use of PRAs had been well established in the nuclear industry and as a result the NRC issued its PRA policy statement directing that the NRC staff use PRA for all regulatory matters to the extent supported.

Probabilistic Risk Assessment represents a comprehensive and disciplined model of plant performance, including interactions between systems and operations. Through the modeling and

subsequent quantification of success and failure paths, a number of potential weaknesses in plant design, operations, surveillance interval testing, and maintenance procedures can be identified. The PRA methodologies can be effectively used within the nuclear industry and supplement engineering evaluation techniques to enhance safety and improve plant availability.

Probabilistic Risk Assessment attempts to quantify the probabilities and consequences associated with accidents and malfunctions by applying probability and statistical techniques and consequence-evaluation methods deemed acceptable amongst technologists. It requires all available information and widens the historic basis by using data not only for accidents but also for plant or equipment failures which have not led to accidents. One of the important features of PRA is the taxonomy and comprehensiveness with which potential accidents are analyzed. In the case of nuclear power, it usually begins with a fault-tree/event-tree analysis of the significant ways in which a plant might fail. A wide spectrum of pertinent accident consequences are then analyzed for potential damage to the public or the plant itself. After the models are developed, the second stage of PRA assigns probabilities of failure for equipment, human actions and their interfaces, and quantifies the consequences of these failures. A measure of the risk associated with the accidents and malfunctions is the curve of probability versus consequences from which various risk parameters can be obtained.

The advantage of the PRA methodology is that it employs a systematic analysis of accident sequences in which system failures, equipment failures, and human errors are considered in the context of their contribution to overall risk. After the accident sequences are constructed, accident probabilities and consequences can be estimated, and contributions to quantitative risk can be measured. Methodologies of PRA have improved the understanding of the effects of multiple, dependent (common-cause) failures on the reliability of nuclear reactor

systems. By using the PRA approach, analysts are able to better understand the contributions of various accident sequences and risks, thus resulting in early identification of additional safety features, nuclear grade equipment, improved reliability, or re-engineering.

The use of PRAs will not eliminate the need for properly designed safety features, the use of engineering safety criteria, safety margins, or Defense-in-Depth. The PRA will however allow nuclear operators to determine weakness within a safety system and properly allocate resources to correct potential safety concerns.

2.1.1 Fault Tree Analysis

Fault Tree Analysis (FTA) was developed by H.A. Watson of Bell Laboratories while in a contract with the U.S. Air Force to study the Minuteman launch control system. Since its discovery in the 60's, FTA has been applied to many applications in the nuclear industry. Traditional FTA has always had a crucial limitation which is the ability to evaluate fault trees when a system is too large and the number of states become overwhelming; much like the massive systems in a nuclear power plant. Limitations in the 60-80's were computer processing speeds. Thanks to the technological advances in microprocessor technology, they no longer pose a problem for this type of analysis [Zouakia et al, 1999].

Fault trees are useful for analyzing complex components and systems, especially in identifying system interrelationships such as shared support systems and common-cause failure mechanisms in highly redundant systems. A fault tree provides a structured approach to determining the probability of failure in a complex system. This approach also illustrates the minimum set of events that can cause the failure of a system. In order to evaluate the industrial risk, it is necessary to have an estimation of the accidents probability. This estimation can be obtained from historical data of previous accidents, or more precisely, from the application of the

FTA. The analysis is restrained in a particular undesired event (accident or incident) defined as the top event. It is operated by means of graphic modeling allowing the visualization of the possible combinations of malfunction and wrong actions that can generate it. The synthesis of the results is generally presented in a graphical model organized by the logic of the Boolean algebra and its symbols [Nivoliannitou et al, 2004].

The basic elements of a fault tree are the same regardless of the types of events or systems being analyzed, therefore a standard terminology and a set of symbols have been developed to represent the events and operations as shown in Table 1[IEEE Std 352, 1987].

Table 1 Fault Tree Symbols [Barlow & Proschan, 1975]

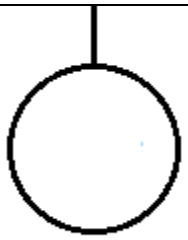
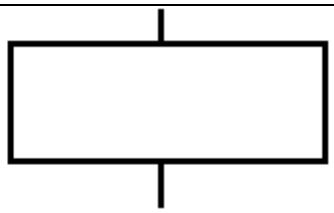
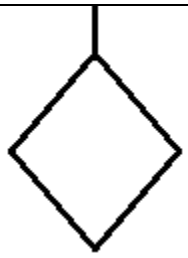
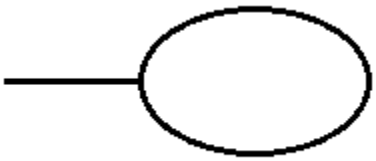
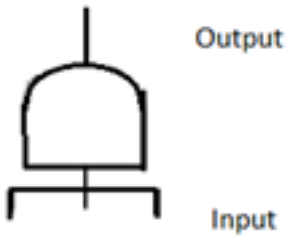
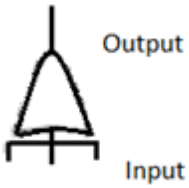
Event Symbols		
	Basic Event	The circle is used to represent basic events in a fault tree. It is the lowest level of resolution in the fault tree.
	Intermediate Event	Top event and intermediate events: The rectangle is used to represent the TOP event and any intermediate fault events in a fault tree. The TOP event is the accident that is being analyzed. Intermediate events are system states or occurrences that somehow contribute to the accident.
	Undeveloped Event	The diamond is used to represent human errors and events that are not further developed in the fault tree.

Table 2 Fault Tree Symbols [Barlow & Proschan, 1975] (Cont'd)

	Conditioning Event	The oval is used to represent a conditioning event – conditions that restrict or affect logic gates
Gate Symbols		
	AND Gate	The event in the rectangle is the output event of the AND gate below the rectangle. The output event associated with this gate exists only if all of the input events exist simultaneously
	OR Gate	The event in the rectangle is the output event of the OR gate below the rectangle. The output event associated with this gate exists if at least one of the input events exists.

The procedure for performing a fault tree analysis consists of the following eight steps [IEEE Std 352, 1987]:

1. Define the system of interest: Specify and clearly define the boundaries and initial conditions of the system for which failure information is needed.
2. Define the TOP event for the analysis: Specify the problem of interest that the analyst will address. This may be a specific quality problem, shutdown, or safety issue.

3. Define the treetop structure: Determine the events and conditions that most directly lead to the TOP event.
4. Explore each branch in successive levels of detail: Determine the events and conditions that most directly lead to each intermediate event. Repeat the process at each successive level of the tree until the fault tree model is complete.
5. Solve the fault tree for the combinations of events contributing to the TOP event: Examine the fault tree model to identify all the possible combinations of events and conditions that can cause the TOP event of interest. A combination of events and conditions sufficient and necessary to cause the TOP event is called a minimal cutset.
6. Identify important dependent failure potentials and adjust the model appropriately. Study the fault tree model and the list of minimal cutsets to identify potentially important dependencies among events. Dependencies are single occurrences that may cause multiple events or conditions to occur at the same time. This step is qualitative common-cause failure analysis.
7. Perform quantitative analysis: Use statistical characterizations regarding the failure and repair of specific events and conditions in the fault-tree model to predict future performance for the system.
8. Use the results in decision-making: Use results of the analysis to identify the most significant vulnerabilities in the system and to make effective recommendations for reducing the risks associated with those vulnerabilities.

2.1.2 Limitations of Fault Tree Analysis

The thesis will only explore the role of traditional FTA and will not consider any dynamic elements. Despite the wide acceptance of fault tree techniques, traditional fault trees are hampered in their application by several problems. Traditional fault tree techniques treat only binary states. That is, fault trees can address only whether the system is completely failed or completely functional. A single fault tree is incapable of addressing degraded states of the system. The traditional fault tree technique is unable to handle any but the simplest models of repair. Fault tree techniques are incapable of handling the continuous change of state of components between failed and un-failed. Fault tree techniques are incapable of adequately modeling sophisticated testing schemes for reactor protection systems. Also, fault trees are considered very complicated and difficult, are time-consuming in its application, and expensive [Marhavila et al, 2011].

2.1.3 Risk Importance Measures

Importance analysis is a part of the system quantification process which enables the analyst to rank the contribution that each component provides to system failure in order to identify the weakest area of the system. Once the weakest area is known, efforts can be concentrated to improve their reliability. Importance measures assign a numerical value between 0 and 1 to each system component; where 1 signifies the highest level of importance. Measures of importance can be categorized as either deterministic or probabilistic. Deterministic measures assess the importance of a component without considering component reliability. A deterministic measure is suited for the development stages when relevant information is limited. Probabilistic measure of importance can be thought of as a measurement for contributing to failure frequency

or contributing to failure probability. An analyst must make the clear distinction of the types of importance measurements which are relevant to their system, equipment or operation. The following are some importance measurements used for PRA:

1. Birnbaum's measure – Birnbaum's measure is the probabilistic measure of component reliability importance. It is the probability that a component is critical to system failure and equals the probability that the system is residing in a critical state for a component such that its failure causes system failure [Birnbaum, 1969].
2. Component Criticality Measure – The component criticality measure is the probability that the component *I* is critical to the system when the component *I* has failed, weighted by the system unavailability at time *t*. These measurements consist of a failure and repair importance.
    a. The failure importance equals the probability that component *I* is fail critical to the system.
    b. The repair importance equals the probability that component *I* is success critical to the system.
3. Fussell-Vesely (FV) – Importance measure of component importance is concerned with component failures contributing to system failure. This measurement is defined as the probability that a minimal cutset containing component *I* can cause the system failure.
4. A component failure can contribute to system failure in the following ways:
    a. Initiating event whereby its occurrence may cause system failure.
    b. Enabling event whereby its existence permits another initiator event to potentially cause system failure.

Initiator and Enabler Importance – Measures of Initiator and Enabler importance analyzes the role of component failures in causing and contributing to system failures. They are both concerned with the sequence of events leading to system failure.

Barlow and Parschan initiator importance is concerned with the failure of components acting as initiating events and thus their occurrence coincides with system failure. This measure calculates the probability that component *I* causes system failure between time at 0 and t [Barlow & Parschan, 1975].

Lambert Enabler Importance is the probability that a failure of component *I* allows system failure between time 0 and *t*, caused by the failure of another component *j* occurring. Where *I* is the enabler and *j* is the initiator [Lambert, 1975].

2.2 Petri Net

Carl Adam Petri defined the Petri net theory in his doctoral thesis titled "Communication with Automata" as a general purpose mathematical tool for describing relations existing between conditions and events [Petri, 1962]. Petri net Theory has developed since and is now considered one of the most powerful tools for the modeling of systems exhibiting concurrency, synchronization, and repair characteristics. As a graphical tool, Petri nets can be used as a visual-communication aid similar to flow charts, block diagrams, and networks. Petri nets have been modified to model and analyze systems in different application areas such as manufacturing, real-time processing, computer architecture, dynamic control, supervisory control, and material handling.

The Petri net is a directed graph consisting of two types of nodes called places and transitions. Weighted and directed arcs connect places to transitions or vice versa. Systems are modeled as a set of conditions and events. Petri nets modeling conditions representing system

states include: top failure state and intermediate system states; which contribute to the accident. Events represent failures which trigger new conditions to occur. Places represent conditions and transitions represent events. Transitions have a set of input and output places which represent the preconditions and post-conditions of the transition. The state of a net is modeled by the presence or absence of a token in the places. The tokens in a place are referred to as the marking of the place. The initial marking represents the initial condition or state of the net. The states change by the firing of transitions which depicts the events occurring. An event occurs only when the preconditions are met and is represented by an enabled transition. The firing of a transition changes the marking of its input and output places, representing a change in its preconditions and post-conditions [Bowden et al, 2000]. A Petri net can be formally defined by the following two methods:

Definition 2.1: (Petri Net) [W. Reisig, 1985] A Petri net is a five tuple structure defined as:

Petri net = $\{P, T, W_{pt}, W_{tp}, M(0)\}$,

where,

$P = \{p_1, p_2, \ldots, p_n\}$- a finite set of places, and $n \geq 0$

$T = \{t_1, t_2, \ldots, t_n\}$ – a finite set of transitions, and $n \geq 0$

There are two weight functions, W_{pt} and W_{tp}, which attach a positive integer weight to each arc of the net connecting places to transitions (pt) and transitions to places (tp), respectively.

The initial marking is represented by $M(0) = [m_1(0), m_2(0), \ldots m_n(0)]^T$ and is a function from the set of places to non-negative integers. The marking at an arbitrary time instant k is represented as $M(k) = [m_1(k), m_2(k), \ldots m_n(k)]^T$ and can also be referred to as the number of tokens in each place.

[Enabled transition]: Transition *t* is enabled if each input place *p* of *t* is marked with at least W(*p*,*t*) tokens. Where W(*p*,*t*) is the arc weight between place *p* and transition *t*.

[Firing of transition]: An enabled transition *t* will fire if the event that it represents occurs. In that case i) *W*(*p*,*t*) tokens are removed from each input place *p* of *t*, and ii) *W*(*t*,*p*) tokens are added in each output place *p* of *t*.

Definition 2.2: Authors have also defined the Petri net as a six tuple structure [T. Murata, 1989]:

PN = {P, T, I, O, M, m_0}

Where,

P = {p1,p2,…,p_n}- a finite set of places, and $n \geq 0$

T = {t1, t2,…,t_n} – a finite set of transitions, and $n \geq 0$

I: PxT→{0,1} – an input incidence matrix that relates places to transitions,

O: TxP→{0,1} – an output incidence matrix that relates transitions to places

M: I, O → {1,2,3,…} – is a weight function

m_0 – The initial marking is represented by $M(0) = [m_1(0), m_2(0), \ldots m_n(0)]^T$ and is a function from the set of places to the non-negative integers. The marking at an arbitrary time instant *k* is represented as $M(k) = [m_1(k), m_2(k), \ldots m_n(k)]^T$ and can also be referred to as the number of tokens in each place.

These two definitions of Petri net are both acceptable and widely used. The conversation equations between the two forms of definitions are:

$$W_{pt} = [W_{pitj}],$$

Where $W_{pitj} = I(P_i, T_j)$ and

$$W_{tp} = [W_{tjpi}]$$

Where $W_{tjpi} = O(T_j, P_i)$. W_{pt} and W_{tp} are called input and output incidence matrices.

[Peterson, 1981][David et al, 1992]

2.2.1 Petri Net Construction

The most recognizable advantage of Petri net modeling is the graphical representation. It allows for users to easily understand and display the formal theory. Places and transitions are represented by circles and bars, respectively. Arcs are shown by arrows and tokens by black dots inside the places. Arc weights are represented by numbers placed by the arc, and an absence of the weight indicates a weight equal to one. The graphical construction of a Petri net involves the understanding of how places, transitions, tokens, and arcs interact.

The following describes in more detail the four basic elements of Petri nets: places, transitions, tokens and arcs [Chiou, 1997].

- ○ : Place, drawn as a circle, denotes a condition in the process.
- □: Transition, drawn as a bar, represents changes in the model. Transitions can be immediate, deterministically time-delayed, or time-delayed based on a probability distribution defined by the user.

→: Arc, drawn as an arrow, determines the path that tokens take throughout the model. Arcs can either enable or inhibit movement in the model depending on their use.

●: Token, drawn as a dot and contained in places, represents objects in the model. In graphical user interface (GUI) applications, tokens are represented as black solid circles and in more sophisticated applications such as color Petri nets; the circles can take on various colors to signify the age. A transition allows the movement of a token and is said to have `fired' when this happens.

Petri nets are a graphical and mathematical modeling tool applicable to many systems. In graphical representation, places represent entities such as conditions, buffers, servers, resources and queues. Transitions represent concepts in real systems such as algorithms and events. Arcs are labeled with their weights (positive integers), where a *k*-weighted arc can be interpreted as a set of *k* parallel arcs.

Labels for unity weights are usually omitted. A marking (state) assigns to each place a non-negative integer. If a marking assigns to place p a non-negative integer *k*, we say that *p* is marked with *k* tokens. Pictorially, we place *k* black dots (tokens) in place *p*. A marking is denoted by *M*, an *m*-vector, where *m* is the total number of places. The p^{th} component of *M*, denoted by *M* (*p*), is the number of tokens in place *p* [Murata, 1989].

The behaviour of many systems can be described in terms of system states and their changes. In order to simulate the dynamic behaviour of a system, a state or marking in a Petri net is changed according to the following transition (firing) rule:

- A transition *t* is said to be enabled if each input place *p* of *t* is marked with at least *w*(*p*,*t*) tokens, where *w*(*p*, *t*) is the weight of the arc from *p* to *t*.
- An enabled transition may or may not fire depending on whether or not the event actually takes place.
- The firing of an enabled transition t removes *w*(*p*, *t*) tokens from each input place *p* of *t*, and adds *w*(*t*, *p*) tokens to each output place *p* of *t*, where *w*(*t*, *p*) is the weight of the arc from *t* to *p*.

n modeling, using the concept of conditions and events, places represent conditions, and transitions represent events. A transition (an event) has a certain number of input and output places representing the preconditions and post conditions of the event

respectively. The presence of a token in a place is interpreted as holding the truth of the condition associated with the place. [Murata, 1989].

2.2.2 Petri Net Interactions

The basic interactions are sequential, synchronization, and merging transitions. The most basic model of the Petri net is the state transition from input place p1 to output place p2. Fig. 1 represents the firing of a token (K) from p1 to p2 based on the transition t1.

From definition the Petri net can be seen as:

The Initial state is: M (0) = [(1, 0)]

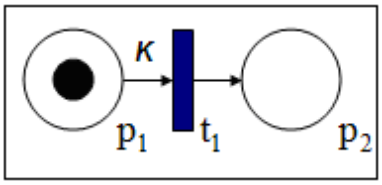


Figure 1 Initial State M(0)

After firing, the state becomes: M (1) = [(0,1)]

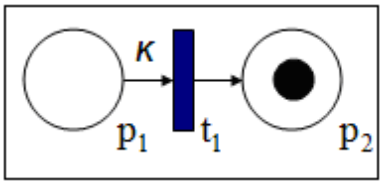


Figure 2 State M(1)

The concept of sequential is shown in Fig. 3. If there is more than one transition in a path, the transition can only fire when its preceding transition has fired. This imposes the precedence of constraint t_2 after t_1.

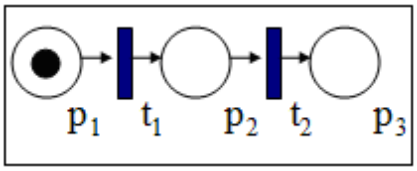


Figure 3 Sequential Interaction

Synchronization is a conditional concept. Previously a transition would fire instantaneously if a place had a token, whereas now the transition is dependent on all connected places having a token. When the transition fires the tokens, the tokens will merge when more than one arrives at the same place. Fig. 4 illustrates synchronization, where transition t_1 fires only when places p_1 and p_2 both have a token. Once the condition is met, t_1 fires, and the two tokens merge. One token is placed in the output place p_3, as depicted in Fig. 5.

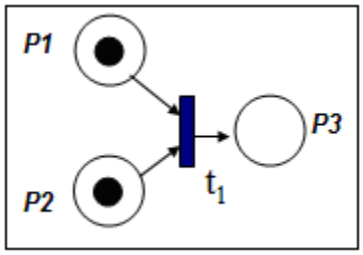


Figure 4 Synchronization Interaction

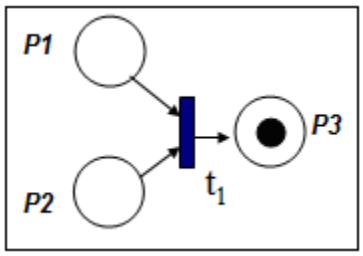


Figure 5 Merged Interaction

2.2.3 Advantages and Limitations of Petri Nets

Using the definition of Petri nets and the concept of sequential, synchronization and merging, the ability to model a system conditional can be achieved.

The advantages of Petri nets are the following [Melnyk, 2003]:

- Petri nets employ local places. Once the model is constructed, users can change the number of tokens or arc weight without modifying the defined conditions.
- Petri nets can model both software and hardware.

- Petri nets model local states instead of global ones. They do not grow out of control as the model increases in complexity.
- Petri nets can be used in various steps of system development and operation, design, testing, and simulation.
- Petri nets allow the user to model dynamic events with distributions other than exponential. This gives the user a more accurate representation when simulating real world processes.
- Petri net simulation allows the user to observe tokens as they move throughout the model in real or simulated time. This simulation feature gives the user a better understanding of the actual process flow and the ability to observe potential conflicts.

Despite these advantages, there are some limitations of Petri nets, as described in the following [Melnyk, 2003]:

- Petri Net software applications are obscure and difficult to integrate with existing software tools. One of the reasons being that developers use different languages for software packages causing integration issues. Also, the steep learning curve associated with Petri net programming is problematic for new users when modifying or designing tools.
- A lack of readily available software packages.
- Only discrete events can be modeled with the basic Petri nets.

2.2.4 Petri Net for Reliability Analysis

System failure analysis based on traditional Petri nets were discussed in [Adamyan & He, 2004], the proposed methods use transitions to identify the sequences of the events in the Petri nets, where the solution is given in terms of the sequences of the firing transition and not the

failed state. To compute the probabilities of sequential failures, data from the firing rates of a transition were used instead of tokens. The methods employs counters, and the number of times transitions are fired to calculate probabilities of sequential failures. This method differs from traditional methods of using markings to determine system failure which can't be applied to sequential failure analysis.

The traditional Petri net is limited to asynchronous systems and it fails to capitalize on modeling of real-time systems. A couple of recognized approaches to overcome this limitation for reliability analysis are Stochastic Petri net (SPN) and Time Petri Net (TPN). Stochastic Petri net is obtained by associating exponentially distributed firing times to the transition. SPN are well suited for performance evaluations.

The Stochastic Petri Net can be defined as a 7-tuple

SPN = { P, T, I, O, M, m_0, Λ }

Where {P, T, I, O, M, m_0} is the marked untimed traditional Petri net and $\Lambda = (\lambda_1, \lambda_2,\ldots,\lambda_n)$is an array of firing rates associated with transitions. A firing delay is associated with each transition. It specifies the amount of time that must elapse before the transition can fire [Ajmone, 1989].

The other approach to address real-time systems is TPN in which, time is assigned to each transition of the original Petri net model replacing instantaneous firing. Another method is to place a delay on the place in the traditional Petri net. It has been shown in [Sifakis, 1980] that assigning time to a transition or place are equivalent, and can be transformed. An alternative method is time Petri net which associates each transition base on an interval $\{t_{min}, t_{max}\}$, which represents the minimum and maximum time during which an enabled transition should fire. [Merlin et al, 1976]

2.3 Chapter Summary

This chapter concludes the literature review which has explored the history of, and introduced the concept of, PRA. PRA was explored in depth along with the most widely used method for modeling and performing PRA, fault trees. The properties and limitations of fault trees are discussed and analyzed. Petri nets are introduced as an alternative modeling tool for PRA. Petri net concepts, construction methods, and their interactions are discussed and examples provided. The advantages and limitations of Petri nets are presented. The following chapter will explore Petri nets in depth. Methodologies are introduced on how to use model Petri nets to provide a structured approach to perform risk assessment.

CHAPTER 3 PETRI NET METHODOLOGY FOR PROBABILISTIC RISK ASSESSMENT MODELING STRATEGY

This chapter investigates the methodologies on how to use Petri nets to model system failures for PRA. The traditional Petri net has a limited capacity to accurately model system failure and therefore perform qualitative and quantitative risk analysis. Gaps between the traditional Petri net and requirements for effective system modeling for PRA are bridged. The behaviour and interaction principles of the traditional Petri net are explored to create transitional logic gates for system failure modeling. A structured approach to Petri net construction is introduced to allow a top down methodology for performing qualitative and quantitative analysis. Petri net software packages are enhanced to incorporate methods to simulate system failure. PRA modeling of systems using Petri net methodology offers distinct advantages over other PRA modeling methods.

3.1 Transformation from Gates to Transitions

In chapter 2, Petri net interactions were briefly discussed and the concepts of sequential, synchronization, and merging transitions were introduced. Building a Petri net model which can accurately model complex systems require knowledge of Petri net behaviours involving multiple initial states, tokens, arcs and simultaneous transitions. Petri net behaviours are explored in the

four cases below and will also act as the building block to develop the transformation from logic gates to Petri net transitions.

The following four Petri net interactions will aid to further enhance the understanding of the basic notions of Petri net construction and behaviours.

In Fig. 6, the initial marking is M_0 = (1, 1, 0), since the tokens are initially in place P1 and P2, with no token in P3. Once the condition of T1 is met, the tokens are fired into P3, removing a token from P1 and P2. Then a single token representing a change in state is fired into P3 with M_1 = (0,0,1) as shown in Fig 7.

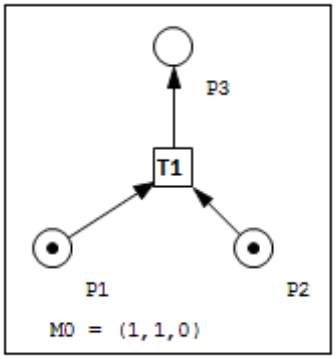


Figure 6 Case 1 Initial State

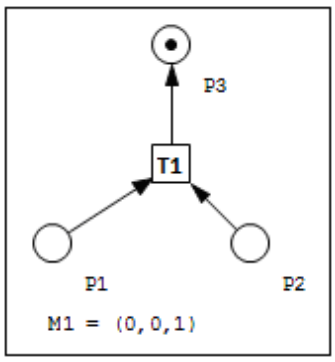


Figure 7 Case 1 State After Firing

In Fig. 8, the initial marking is M_0 = (2, 1, 0), since the tokens are initially in place P1, and P2, with no token in P3. Once the condition of T1 is met, the tokens are fired into T1, removing a token from P1 and P2. Then a single token representing a change in state is fired into P3, while a token still remains in P1. Therefore the state after firing is M_1 = (1,0,1), as shown in Fig 9.

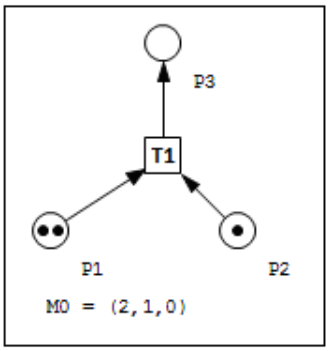


Figure 8 Case 2 Initial State

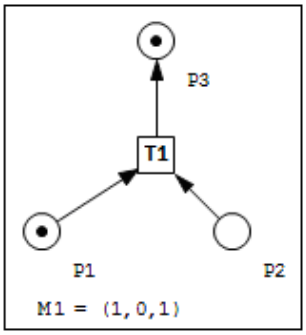


Figure 9 Case 2 State After Firing

In Fig. 10 the initial marking is M_0 = (1, 1, 0, 0), since the tokens are initially in place P1, and P2, with no tokens in P3 or P4. Once the condition of T1 is met, the tokens are fired into T1, removing a token from P1 and P2. Then a single token representing a change in state is fired into P3 and P4, therefore the state after firing is M_1 = (0,0,1, 1) as shown in Fig 11.

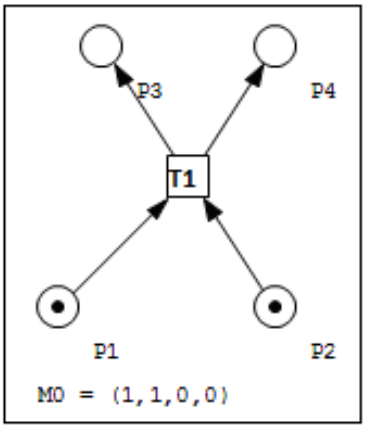


Figure 10 Case 3 Initial State

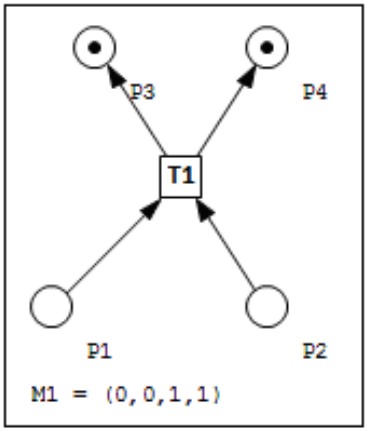


Figure 11 Case 3 State After Firing

In Fig. 12 and Fig. 13, two transition conditions are available T1 and T2. Unlike the previous examples, the transition firing condition is not dependent on P1 and P2 having tokens simultaneously. Instead, a token in P1 will satisfy the condition to fire tokens from T1 into output places P3 and P4.

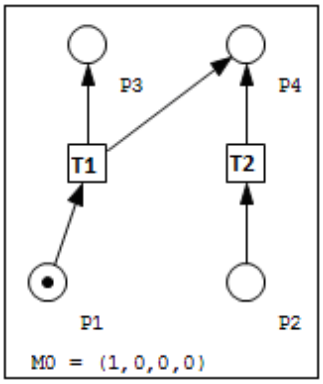


Figure 12 Case 4 Initial State

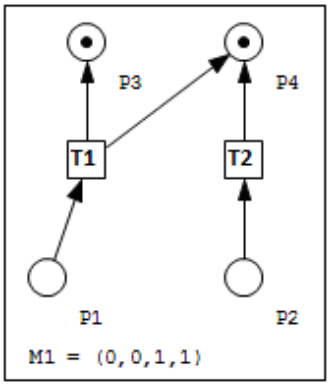


Figure 13 Case 4 State After Firing

Petri nets can also be thought of as a series of Boolean functions, 1 and 0, where 1 represents a present state and 0 a non-active state. If we continue to think of a Petri net in terms of Boolean functions, then for every transition, a truth table can be generated.

Using the system in Fig. 14, the transition T1 firing is conditional based on places P1 and P2 both having a token. The truth table that we can deduce from this Petri net is shown in Table 2.

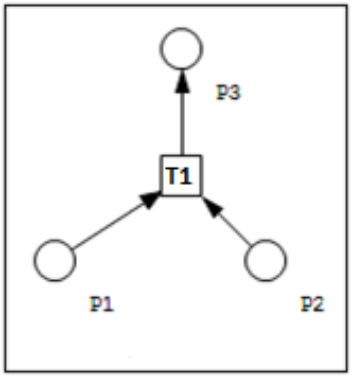


Figure 14 AND Gate

Table 3 Truth Table of AND Logic

Input		Output
P1	P2	P3
0	0	0
0	1	0
1	0	0
1	1	1

This Petri net's truth table is exactly the same as an AND gate logic. Therefore the Petri net from Fig. 14 can be represented as the logic gate AND. Another logic gate OR can also be represented by manipulating the Petri net interactions to match their respective properties. Using the system in Fig. 15, transition T1 and T2 firing condition's based on places P1 or P2 containing a token.

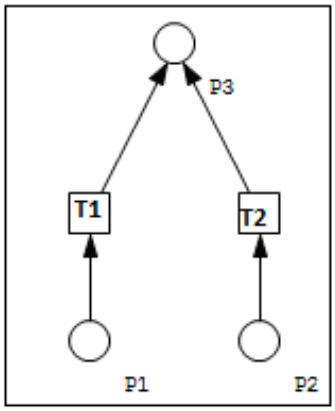


Figure 15 OR Gate

Table 4 Truth Table of OR Gate

Input		Output
P1	P2	P3
0	0	0
0	1	1
1	0	1
1	1	1

The fact that Petri nets have the ability to be interpreted as a set of Boolean functions allow Petri nets to be a viable alternative to event tree and fault tree modeling techniques. Also, the use of Petri net logic gates will allow qualitative analysis techniques such as the top down approach to be used to obtain minimal cutsets and path sets as seen in later sections. For the purposes of this thesis, Petri nets will represent modeling based on system failures and top events.

Now that a set of Petri net logic gates have been established, one can model systems or transform existing systems modeled by fault trees and event trees. The following is to demonstrate converting a fault tree into an equivalent Petri net. Figure 16 is a fault tree created using the software Relex to model the logic control system of a set of transmitters and then converted to its equivalent Petri net model, as shown in Fig. 17 to demonstrate the techniques shown previously for AND and OR gate transformations. A series of voting gates represented by G1, G2, and G3 are present to indicate when a circuit redundancy has been exceeded, and to annunciate or trip a system. Voting gates are widely used in the nuclear systems for logic controls. The figures use general basic events B1, B2, B3, B4, and B5 to represent system state failures of transmitter communication including low pressure indication, low flow indication, and insufficient or excessive level indications.

Working from node G1, we notice that G1 is an AND gate and will become true only when basic event B1 and node G4 are true. G4 is a subset of G1 and it is an OR gate which is true if any of its inputs B4 or B5 is true. The transformation of gates start from bottom up, therefore we can start with G4.

Using Fig. 15 as a reference, we notice that basic events B4 and B5 correspond to P1 and P2 respectively. The OR gate will be modeled by two basic event places, (B4 & B5), each with an arc to their transitions and an arc from the two transitions to place G4.

Then we move on to node G1, which is an AND gate. Using Fig. 14, we notice that B1 and the node G4 (transformed above) are identical to P1 and P2 respectively and therefore can be represented by one transition and two places, (B1 & G4), with arcs pointing to the transition and another arc pointing to place G1.The rest of the transformation can be interpreted in a similar manner.

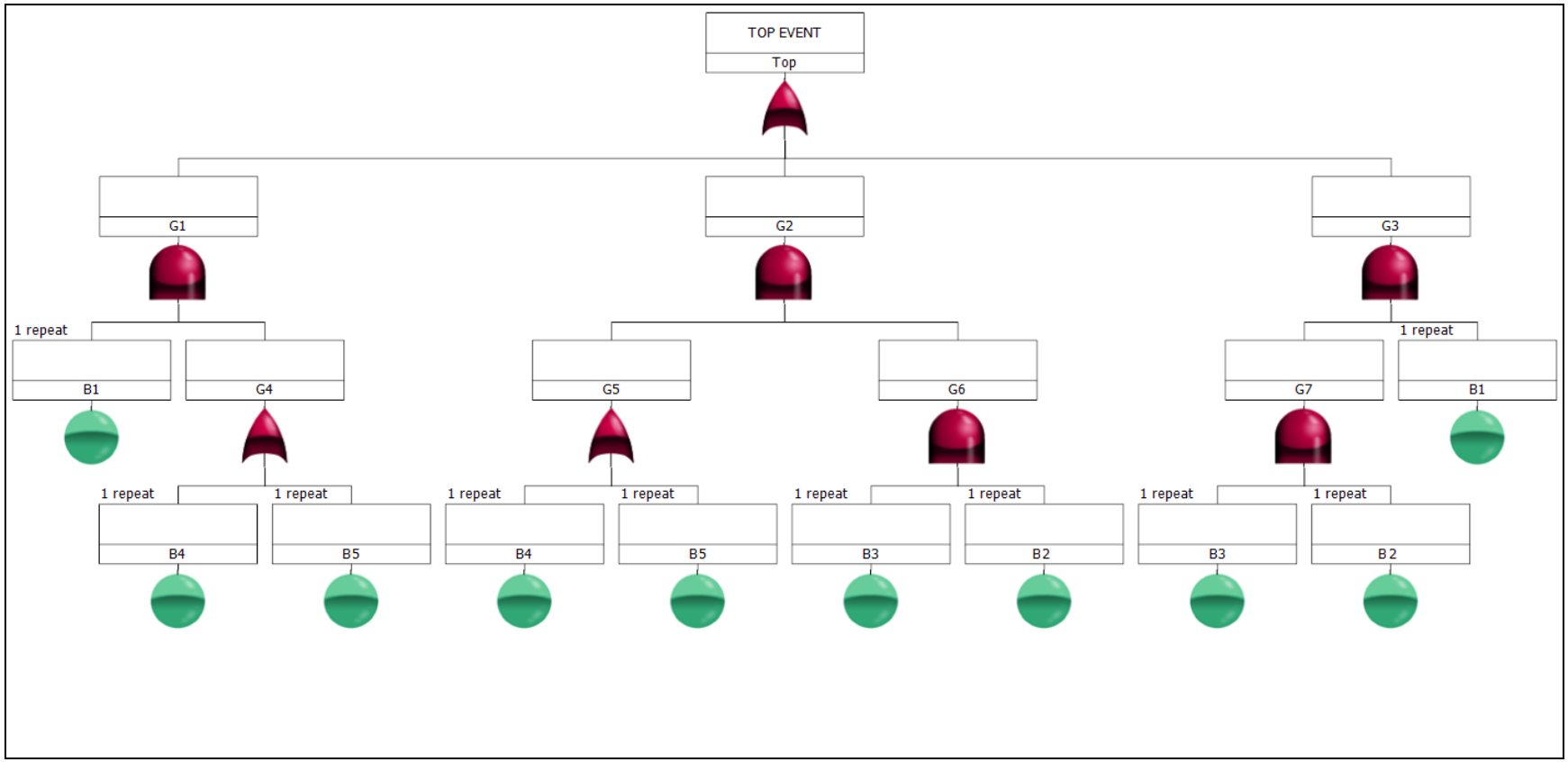


Figure 16 Fault Tree: Voting Gate

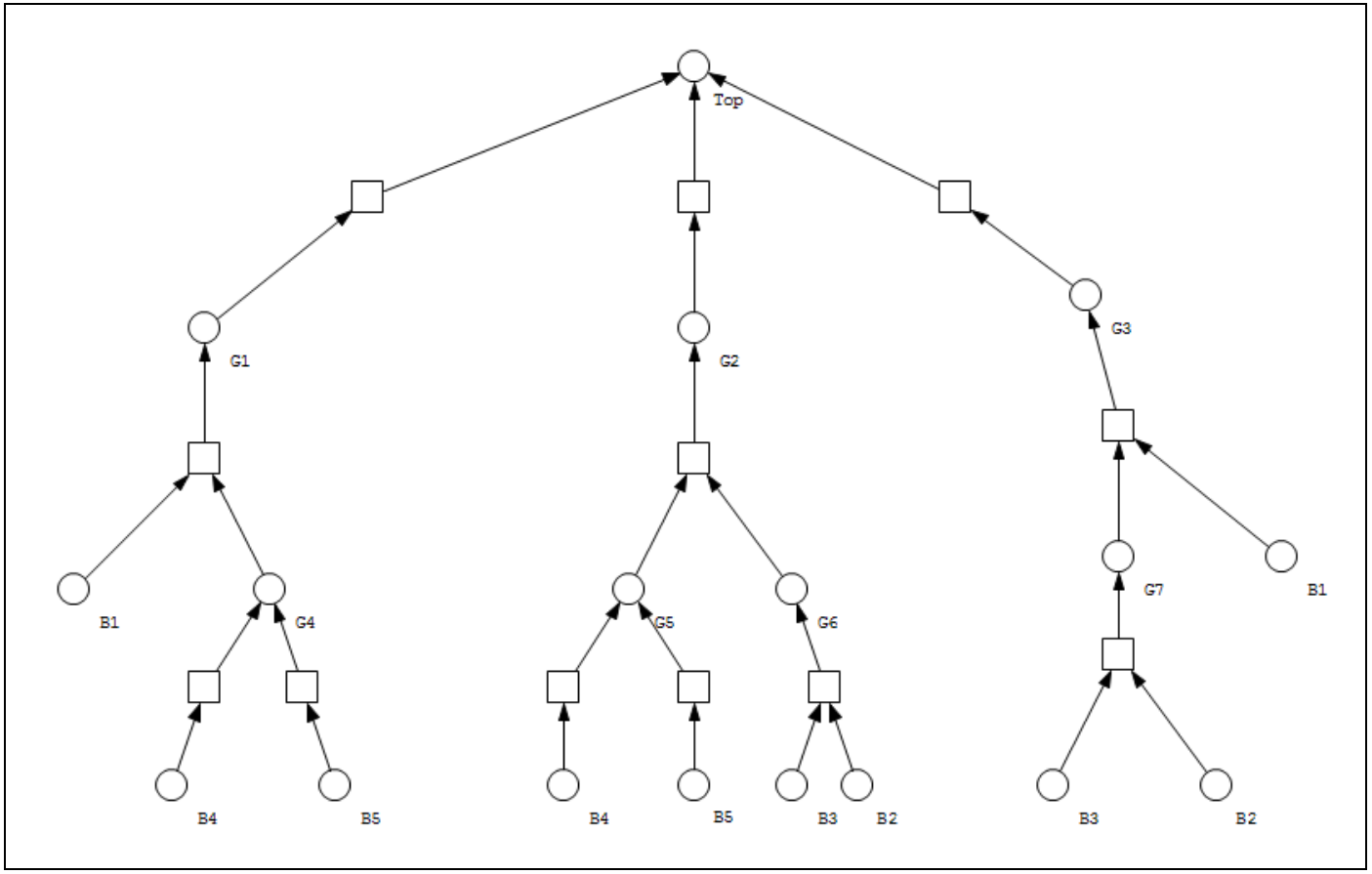


Figure 17 Petri net: Voting Gate

3.2 Reachability Analysis versus Top Event Analysis

Petri net analyses are performed to determine system properties such as: reachability, liveliness, safeness, boundedness and stability, conservation, and controllability. The main approach for this kind of behavioural analysis is to use the reachability concept.

Reachability is the most basic problem of a Petri net analysis. A marking is reachable if there exists a sequence of transition firing which, starting at the initial marking, results in that marking.

A net is called live if all transitions in that net are potentially fireable. This is of interest to resource allocation system modeling, where avoidance of deadlocks is important. A deadlock in a Petri net is defined as a transition or a set of transitions which cannot fire.

In the Petri net theory, a safe place can have only zero or one token at a time. This property is used in fault detection to check if a place has erroneously acquired more than one token when it was meant to be a safe place.

Boundedness examines whether one or more of the system states can grow beyond a limit or bound. Stability checks a system's bounded inputs to determine whether its outputs would continue to remain bounded. When the total token count in a set of Petri net places remains constant, those places are called conservative. This property is useful in modeling resource allocation in a system.

The controllability of a system is defined as an answer to the question "Is it possible to steer a system from a given initial state to an arbitrary state in a finite time period?" [Ogata, 1987]. A Petri net is said to be completely controllable if any marking is reachable from any other marking [Murata, 1989].

The reachability tree method is based on constructing a complete reachability tree or a subset of all reachable states. The reachable states found by this method depend on the initial

marking of the net. The reachability tree can help users analyze behaviours of the system or structure. The advantage of a reachability tree is that it can be applied to any system, and not limited to only sub-class nets [Reisig, 1992][Lutenbach, 1987].

The construction of a reachability tree begins with the marking M (0) as the root node of the tree which discovers other nodes by firing all enabled transitions. Arcs represent the transition firings and reveal the nodes which are reachable from the transition. Each new node will either be a reachable node or a dead end. The tree is complete when all reachable states are discovered.

The limitations of the reachability tree are that it is only useful for small nets since the reachability tree size explodes rapidly with an increase in the state space size.

3.2.1 Marking Transformation

The state of a Petri net is represented by marking M. The K^{th} state M_k determines the next state M_{k+1}. [Murata, 1989].

$$M_{k+1} = M_k + A^T S \ (k = 0, 1, 2,..., n.) \qquad (3.1)$$

Where,

$\mathbf{M_k}$ is a column vector whose k^{th} component is the marking of place P_k .

$\mathbf{A^T}$ is an incidence matrix whose rows are associated with places, and columns are associated with transitions.

S represents a column vector whose i^{th} component denotes the firing time of T*i*.

Combining all marking transformations from an initial marking M_o to a final marking M_n, Eqn. 3.1 can be rewritten as:

$$M_n = M_o + A^T \sum \qquad (3.2)$$

and,

$$A^T \sum = \Delta M = M_n - M_o \quad (3.3)$$

Where $\sum$ denotes a firing-count vector, i.e. assume a firing sequence of T2, T3 and T3 then the resulting firing-count vector $\sum_1 = [01101]$.

Using the marking transformation and reachability concept, Petri net analyses can be used to determine system failure and the events leading up to the system failure. Whenever a token enters the top place, it indicates a system failure. Therefore, by setting the last component marking column vector to be the top place, $M_n = [**,,,*1]^T$, where *n* denotes the n^{th} modification and * represents the number of tokens in the n^{th} place, the events leading up to the system failure can be determined.

3.2.2 Renumbering Matrix Method for Static Analysis

The marking transformation may be prone to human error when analyzing a large system complex system, therefore the renumber matrix method was developed to ease the difficulty of keeping track of places and transitions as the number of places and transitions were not always equal. The renumber matrix method addresses the issue by reorganizing the incidence matrix to a triangle matrix. The resultant matrix simplifies the transformation analysis and becomes user friendly. The method used to modify the existing Petri net is as follows:

1. Assign numbers to basic places.
2. The numbers of places and transitions are the same. If there are multiple input places connected to a common transition, the number of each transition has as many characters as the number of input places.
3. Number other output places and transitions.
4. Put the numbers in entries of an incidence matrix.

5. Append a column with its last entry as -1 to the right of the matrix. A square matrix is thus formed.

To demonstrate this renumbering method, the Petri net, which represents a 2-out-of-3 voting logic as shown in Fig. 18, will be modified. Its associated triangular matrix is presented in Table 4. 2-out-of-3 voting logic gates are found in various redundant systems within a nuclear power plant including Shutdown System 1 (SDS1), annunciations, and Instrument and Control (I&C) logics to name a few. After applying the renumbering algorithm, the modified Petri net, Fig. 19, is obtained. Notice that the Petri net system remains unchanged and all logic gates are still identical to Fig. 18. This was simply a nomenclature modification, where the number of terms for each transition matches the places.

In the following reachability model, it can be said that if the initial state M_0 can reach place P_{18} by M_n states, then the top event is reachable. This is demonstrated using an initial marking $M_0 = [0100000010000000000]^T$.

All arc weights for this system are 1. T2 is enabled and can fire since P_2 has a token.

Therefore $M_1 = [0000000011000000000]^T$.

The transition gates T9_T10, which represents an AND gate, is enabled when both places P9 and P10 contain a token. When the AND gate transition is enabled, both tokens in P9 and P10 are removed and then fired into P_{15}.

Therefore $M_2 = [0000000000000001000]^T$.

Finally, transition T15 is enabled and can fire which leads to the top event P18 becoming active and reachable as shown in M_3. Once the marking M_3 event occurs, the end of the tree has been reached.

$M_3 = [0000000000000000001]^T$

Using the static analysis method, the reachability concept, it has been determined that with an initial marking of M_0, the top event P18 can be reached within M_3. The renumbering method of Petri net construction has simplified the configuration for places and transitions to perform analysis for static analysis. This method also allows users to build an *n* x *n* matrix to trace and track Petri net systems for system failure modeling analysis.

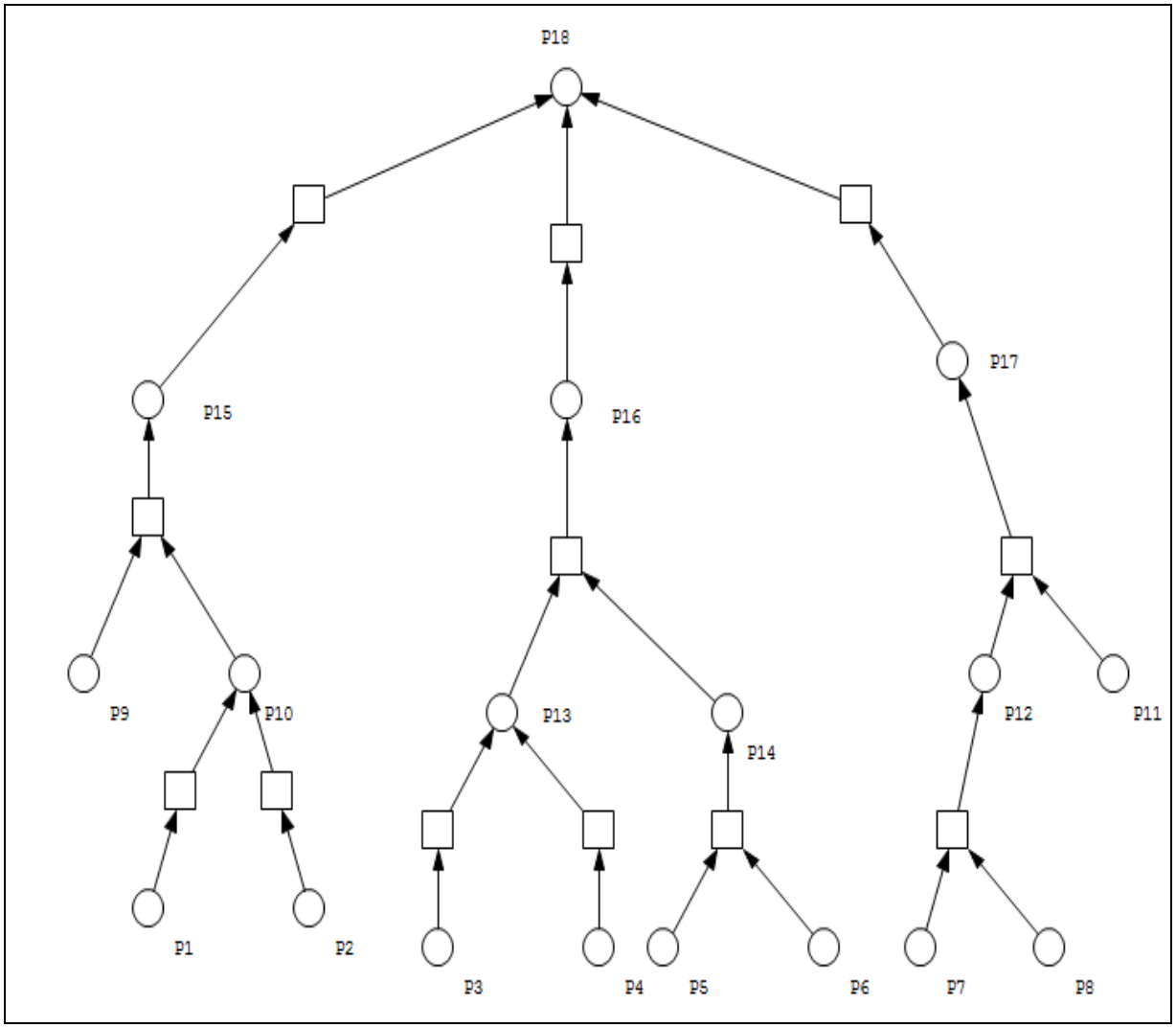


Figure 18 Petri net: 2-out-of-3 Voting Gate

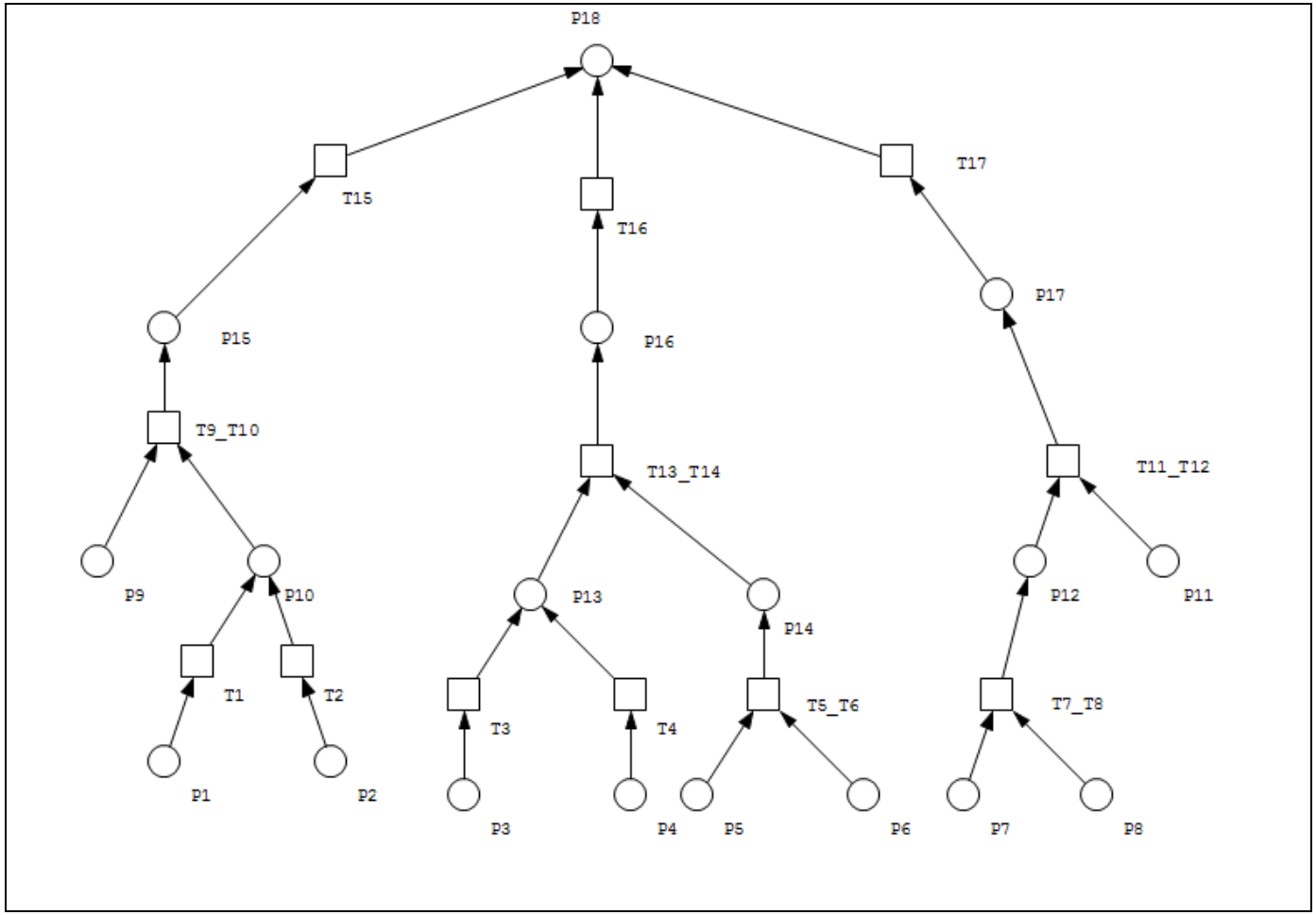


Figure 19 Renumbered Petri net of Figure 18

Table 5 Renumbered Matrix of Figure 19

A^T =

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18
P1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P3	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P4	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P5	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
P6	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0
P7	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0
P8	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
P9	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0
P10	1	1	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
P11	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0
P12	0	0	0	0	0	0	1	1	0	0	0	-1	0	0	0	0	0	0
P13	0	0	1	1	0	0	0	0	0	0	0	0	-1	0	0	0	0	0
P14	0	0	0	0	1	1	0	0	0	0	0	0	0	-1	0	0	0	0
P15	0	0	0	0	0	0	0	0	1	1	0	0	0	0	-1	0	0	0
P16	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0
P17	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	-1	0
P18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	-1

3.3 Minimal Cutset Analysis Using Petri Net

Qualitative analyses are used to identify possible ways in which a system can fail and to identify proper precautions (design changes, administrative procedures, etc) that will reduce the frequency or consequences of such failures [IEEE Std. 352-1987]. A qualitative reliability analysis can be performed with one or more of the following objectives:

1. To identify weak spots or imbalances in the design.
2. To aid in the systematic assessment of overall plant safety.
3. To document and assess the relative importance of all identified failures.
4. To provide a systematic compilation of data as a preliminary step to facilitate quantitative analyses.

One of the most important results from qualitative analyses is the minimal cutset. Minimal cutsets describe the combination of events or component failures that cause the top event to occur i.e. system failure. From Section 3.1, it was determined that Petri nets could represent a set of Boolean functions. Therefore, it could model system failures for PRA similar to fault trees. When system failure is modeled as a set of Petri net logic gates, the top-down methodology, which is used to solve for MCS, becomes an easy and viable method for qualitative analysis. The method is described as follows [Murata, 1989]:

1. Write down the numbers of places by making a horizontal arrangement if the output place is connected by multi-arcs to transitions.
2. Write down the number of places by making a vertical arrangement if the output place is connected by an arc to a common transition.

3. When all places are replaced by basic places, a matrix is established. If there is a common entry located between rows or columns, it is the entry shared for each row or column. The column vectors of the matrix represent cutsets while row vectors represent path sets.
4. Remove the supersets to obtain the minimal cutsets and minimal path sets.

To demonstrate the top-down methodology, Fig. 17 is used. Table 5 represents the four steps to determine the minimal cutsets.

Step 1: The horizontal arrangement represents the events to the top event, which are G1, G2, and G3.

Step 2: Each event can be reduced to a set of places, events, or a combination of places and events. In the case of event G1, it can be reduced to place B1 and event G4, where G1 is true when the AND logic between B1 and G4 is true. G4 can be further reduced to the place B4 and B5 where G4 is true when the OR logic of B4 or B5 is true. The same steps are applied for G2 and G3.

Step 3 is represented by Table 5.

Step 4: Identify the minimum cuts set(s).

Table 6 Minimal Cutset for Figure 17

Step 1	G1	G2	G3
Step 2	B1 G4	G5 G6	B1 G7
	B1 B4B5	B4 B5 B3 B2	B1 B3 B2
Step 3	B1 B4B 5	B4 B5 B3 B2	B1 B3 B2
Step 4	[B1, B4], [B1, B5], [B2, B3, B4]. [B2, B3 B5], [B1, B2 B3]		

Therefore the results for qualitative analysis for minimal cutsets using the top down methodology are as follows: [B1, B4], [B1, B5], [B2, B3, B4]. [B2, B3, B5], and [B1, B2, B3]. The minimal path sets can be obtained with a similar method to the minimal cutsets. Step 1 continues the events that lead to the top event in a horizontal arrangement, are seen in Fig. 17, G1, G2 and G3. Identical to Step 2 from the MCS analysis, the events can be reduced to a set of places or combination of places and events. As seen in Table 6, G1 is reduced to place B1 and event G4 and further reduction of G4 becomes events B4B5. The same method can be applied to G2 and G3 to obtain their places. Step 3 in Table 6 represents the places of each event and the expanded Step 3 below represents all possible combinations of places. For example, the first entry B1 B4 B5 B1 is obtained by the places B1, B4B5 from G2 and B1 from G3 event. The second entry B1B4B5B3 is obtained by the place B1, B4B5 and B3 from event G7 and the third entry is obtained by the places B1, B4B5 and B2 from event G7. The same step is applied to all the other places, and will be expanded to represent an exhausted list of combinations for events G1, G2 and G3. Once all path sets are obtained, the minimum path set can be selected which are: [B1, B3], [B1, B2], [B1, B4, B5], [B3, B4, B5], [B2, B4, B5]. The traditional methods for qualitative analysis do present a challenge to human error due to volume but, there are mechanisms in place such as the renumbering matrix to reduce the likelihood of error.

Table 7 Minimum Path Set for Figure 17

Step 1	G1	G2	G3
Step 2	B1 G4	G5 G6	B1 G7
Step 3	B1	B4B5	B1
	B4B5	B3	B3
		B2	B2
Step 4	[B1, B3], [B1, B2], [B1, B4, B5], [B3, B4, B5], [B4, B5, B2].		

3.4 Non-coherent System Analysis

Non-coherent events are encountered in nuclear safety systems due to the regulatory requirement that not all redundant loops should be simultaneously disabled due to maintenance. In a non-coherent system, both components' failed and non-failed states are events for system failure analysis. Non-coherent systems are represented in a fault tree by a NOT gate, or exclusive OR (XOR) gate. The use of the NOT gate increases the complexity of analysis.

Once the NOT gate is used, cutsets do not apply any more since the fault tree no longer has monotone properties (the definition of non-coherent fault trees can be found in Appendix D). Therefore the minimum cutset concept should be replaced by a set of literals in a prime implicant in Boolean algebra. A complete set of prime implicant sets gives all modes of system failure so that qualitative and quantitative system analysis can be completed. However, this is rarely done in practice due to time consuming calculations. In nuclear systems, separate PRA models are developed and analyzed for each redundant loop. Each event is then subsequently used as an input in an OR gate for analysis.

Non-coherent systems can be modeled with Petri nets as well with the introduction of the inhibitor arc, as shown in Fig. 20 below. The inhibitor arc is shown as a line with a hollowed out circle at the edge. This edge is equivalent as a NOT gate at the end of an arc.

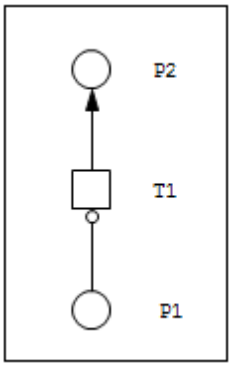


Figure 20 Inhibitor Arc

Consider the non-coherent fault tree in Fig. 21, whose top event is represented as an AND combination of *n* monotonic sub-trees.

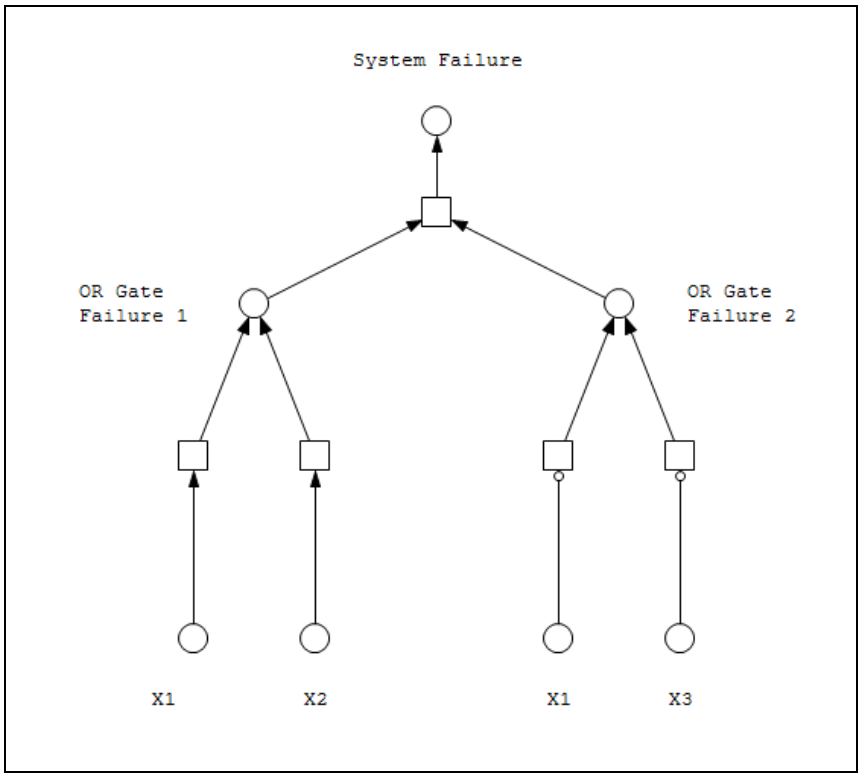


Figure 21 Non-Coherent System

According to the definition of monotonic fault trees, the derivation of minimal cutsets is equivalent to that for coherent fault trees. Failure 1 from Fig. 21 can be solved with the traditional top-down algorithm, and Failure 2 from Fig. 21, can be obtained as follows:

Let Y denote the binary indicator variable for system failure in Fig. 21. Prime implicant can be obtained as follows:

$Y = (X_1 + X_2) * (X_1' + X_3')$

$= (X_1 * X_1') + (X_1 * X_3') + (X_2 * X_1') + (X_2 * X_3')$

$= (X_1 * X_3') + (X_2 * X_1') + (X_2 * X_3')$

Where x_n' refers to the nonexistence of event x_n.

Therefore, prime implicants can be obtained as $\{X_1, X_3'\}$, $\{X_2, X_1'\}$, and $\{X_2, X_3'\}$.

3.5 Quantitative Risk Analysis

Probabilistic Risk Assessment can be split into two stages: qualitative analysis and quantitative analysis. In a quantitative analysis, users represent the system by a mathematical model, assign probabilities to each failure mode of concern, and reconcile the calculated estimates of system unavailability and unreliability with the overall system goals.

System availability ($A_s(t)$) is the probability that the top event does not exist at time *t*. This is the probability of the system operating successfully. On the other hand, System unavailability ($Q_s(t)$) is the probability that the top event exists at time *t*. This is either the probability of system failure or the probability of a particular system hazard at time *t*. It can be seen that the system unavailability is complementary to the availability, and the following identity holds:

$$A_s(t) + Q_s(t) = 1 \quad (3.4)$$

Other variables relating to quantitative analysis include system reliability, unreliability, failure density, conditional failure intensity, unconditional failure intensity, and mean time to failure. For the purposes of this thesis, the focus will be system unavailability.

The assumption regarding basic events P_1, ..., P_n is that they are independent, which means that the occurrence of a given basic event is not affected by the occurrence of any other basic event. Therefore,

$$Pr\{P_1 \cap P_2 \cap ... \cap P_n\} = Pr\{P_1\}Pr\{P_2\}...Pr\{P_n\} \quad (3.5)$$

Where the symbol $\cap$ represents the intersection of events

Consider an AND gate where simultaneous existence of basic events result in the top event. The system unavailability $Q_s(t)$ is given by the probability that all basic events exist at time *t*:

$$Q_s(t) = Pr\{P_1 \cap P_2 \cap ... \cap P_n\} = Pr\{P_1\}Pr\{P_2\}\ldots Pr\{P_n\} \quad (3.6)$$

Consider an OR gate where the top event exists at time *t* if, and only if, at least one of the *n* basic events exists at time *t*. The system unavailability $Q_s(t)$ is given by:

$$Q_s(t) = Pr\{P_1 \text{ U } P_2 \text{ U... U } P_n\} \quad (3.7)$$

Where the symbol U denotes a union of the events

From Eqn. (3.4),

$Q_s(t) = 1 - A_s(t)$

$= 1-[1-Pr\{P_1\}][1-Pr\{P_2\}]...[1-Pr\{P_n\}]$ (3.10)

It is possible to describe the state of the basic event or the system by a binary indicator variable. If we assign a binary indicator variable Y_i to the basic event *i*, then:

$Y_i = 1$, when the basic event *i* exists

$Y_i = 0$, when the basic event *i* does not exist

Top event is associated with a binary indicator variable Ψ (Y) related to the state of the system by:

$\Psi(Y) = 1$, when the top event exists

$\Psi(Y) = 0$, when the top event does not exist.

3.6 RELEX Software Application

For PRA, it is quite inefficient to have users model and calculate qualitative and quantitative analysis by hand. The modeling alone would be too time intensive, and the likelihood of human error is great. Therefore, using computer software to model and perform calculations is often used even for the most rudimentary tasks because they allow users to make instant modifications.

In general, a PRA software tool should be able to accomplish the defined PRA tasks. Some tasks in particular identify initiating events, model scenarios using fault tree or event sequence diagrams, accommodate multiple mission stages, model failure mechanism using fault trees, quantify the scenarios and fault trees using Boolean reduction, perform uncertainty, sensitivity and ranking analysis, and provide the correct results.

Relex provides a set of reliability analysis tools including Reliability Prediction, Reliability Block Diagram (RBD), high level Failure Mode and Effects Analysis (FMEA or FMECA), Fault Tree, Failure Reporting, Analysis, and Corrective Action System (FRACAS), Event Tree, and more. In this thesis, Relex Studio fault tree components are used. The software program chosen to meet the demands of such assessment is Relex [Relex, 2007-2012].

3.7 Petri Net Software Application

The Petri nets' most visible advantage over traditional PRA modeling techniques such as fault tree analyses, is the ability to be used as a graphical tool to simulate system and process, providing users with a more hands-on approach to modify and analyze system behaviour. For the purpose of this thesis, the Petri net software used is Snoopy [Rohr et al, 2010]. Snoopy has the ability to model and simulate system modeling for traditional Petri nets, extended Petri nets, and extended stochastic Petri nets. The simulation feature allows user to start, pause, and modify places and tokens during mid process. Snoopy has the following distinguished features:

1. It is extensive. Its generic design facilitates the implementation of new graph types.
2. It is adaptive by supporting the simultaneous use of several graph types, while the GUI adapts dynamically to the graph type in the active window.

For the purposes of PRA, the objective is to determine system failure, the occurrence of top events. Petri net simulation currently involves manually inputting tokens into places (initial marking), and examining their behaviour in a system. In the case of system failure analysis, this is the reachability concept. However, there are two major limitations of Petri net simulation for PRA:

The first limitation is that Petri net simulation does not recognize that each Petri net place (event), for the purposes of PRA system modeling, should represent a Boolean state. The state should indicate to the user whether it has failed or not. This limitation exists because of the definition of Petri nets Eqn. (2.1) where the weighted function $M(k) = [m_1(k), m_2(k), \ldots m_I(k)]^T$ refers to the number of tokens in each place.

The second limitation with Petri net software is the ability to simulate a system failure without any user initial markings. Initiating events which do not require user initial markings input ensure a more random simulation.

Snoopy is not open source and therefore does not provide source code to be modified to overcome the limitations for PRA modeling purposes. Therefore a new software package is developed, introduced, and explained in detail to overcome the limitations of modeling for PRA within the Snoopy software. The software package will consist of Java classes for places, arcs, and transitions for a traditional Petri net. It should be noted that the software package is not meant as a stand-alone program to model and simulate Petri net but rather an interpretation of the Snoopy source code with enhancements to overcome the two limitations of a failed state and random initial markings.

3.**7.1 Petri Net Failed State Simulation**

When the Petri net definition is modified such that a place can consist of a Boolean state, this will translate into a more accurate modeling technique for the purposes of PRA where the main concern is the probability of the top event occurrence. The place marking will still accept tokens, and transitions will continue to be enabled based on conditions set out by the arc weight. Once the conditions of the transition are enabled, the place will then be in a failed state and will not be able to accept further tokens. This thesis does not explore system behaviour when repairs are introduced; therefore once a place's state is failed, it will remain failed until a new simulation has begun.

The main software classes required to develop a traditional Petri net are place, transition, and arc classes. A place may have an integer or Boolean initial marking, an integer weight, input

transitions, and output transitions. The transitions must have valid transition identifiers at the point of creation. During place creation, users must provide names to place(s) with a given Petri net system and define their attributes. The following properties belong to the place class:

PlaceCreation	‘place’ Place
	‘place’ Place PlaceDefinition
Place	Identifier PlaceParams
PlaceParams	(InitialMarking)
	(UpperBound)
	(InitialMarking , UpperBound)
InitialMarking	IntegerExpression
UpperBound	IntegerExpression
	BooleanExpression
Integer Expression	Constant or,
	IntergerVariable
IntegerVariable	Identifier
BooleanExpression	BooleanVariable
BooleanVariable	Identifier
PlaceDefine	PlaceProperty
PlaceProperty	PlaceInputs or,
	PlaceOutputs
PlaceInputs	‘in’ ‘:’ TransitionList
PlaceOutputs	‘out’ ‘:’ TransitionList
TransitionList	TransitionArc

TransitionArc	TransitionName ArcWeight
TransitionName	Identifier
ArcWeight	IntegerExpression

Transition creation function provides names to transitions in the Petri net and defines their attribute. Transitions may consist of input places, output places, and conditions for firing. Input and output place names must be valid identifiers at the point of the creation of the transition. The conditions of firing are given the attribute name ‘fire’. The values of this attribute are immediate, or FireCondition. Immediate attribute indicates the immediate firing of a token once the conditions of a transition are true. The immediate attribute is currently the only attribute available for traditional Petri net simulation using the Snoopy software. The FireCondition is a public method. When called, it will block the firing until the transition condition(s) are enabled, before firing the transition. The following properties belong to the Transition class:

TransitionCreation	‘transition’ Transition
Transition	Identifier
TransitionDefine	TransitionAttribute + FireCondition
Expression	Constant or, Variable
Variable	Identifier
TransitionAttribute	TransitionInputs TransitionOutputs FireCondition
Transition Inputs	‘in’ ‘:’ PlaceList

TransitionOutputs	‘out’ ‘:’ PlaceList
FireRule	‘fire’ ‘:’
	Immediate or,
	fireCondition
PlaceList	PlaceArc
PlaceArc	PlaceName + ArcWeight
PlaceName	Identifier

Now that the classes for place and transition are defined within a Java environment, Petri net coding can be used to define Petri net systems.

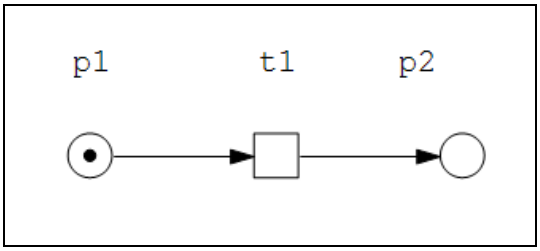


Figure 22 A Simple Petri Net

Figure 22 represents a simple Petri net. There are two places, *p*1 and *p*2, and one transition, *t*1. The initial marking is one token in *p*1. *t*1 has one input arc from *p*1 and one output arc to *p*2. *t*1 is currently enabled because *t*1’s input place (*p*1) has a token, and is ready to fire. The arc weight of Fig. 22 has a default value of 1. The transition *t*1 will fire because all the conditions for firing have been met: a token available in the transitions input place and the token is equal to or greater than the weight of the arc connecting it. To represent this Petri net in Java language, the user might use a specification such as:

```
net SimplePetriNet
place p1(=1), p2;
transition t1 {
in:p1;
out: p2;
}
```

In the code above, SimplePetriNet defines the two places (*p*1 and *p*2), and a transition (*t*1), as seen in Fig. 22. Input (in) for place (*p*1), where *p*1(=1) indicates that *p*1 has an initial marking of one token. Once the program runs, SimplePetriNet will look at the conditions of *t*1 and notice that it is enabled and fire it. This will cause the token in *p*1 to be removed and place a token into *p*2 as shown in Fig. 23.

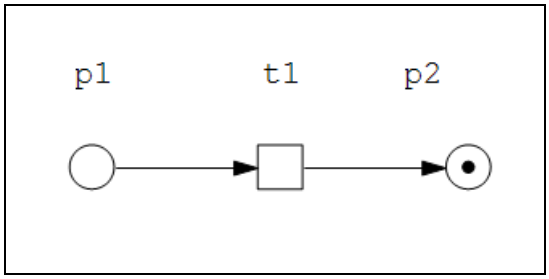


Figure 23 A Simple Petri Net Transition

The Petri net model in Fig. 22 is an example of the immediate firing condition. Sometimes, the user will want to control the time when a transition fires when all conditions are met and the transition is enabled. In this case, a slight modification to the code is required; the transition property 'fire' is changed to fireCondition, as follows:

transition t1 {

in: p1;

out: p2;

fire: fireCondition

}

In this case, transition *t*1 will not fire until it is requested to do so by the user.

The limitation of the current Petri net software is the inability to simulate a modeled system for the purpose of PRA, because the current Petri net simulation structure cannot properly represent a failed state once the subsequent transition fires. In Fig. 22, the current place *p*1 has a token, which represents a failed state. But once the transition *t*1 fires, the token is removed from *p*1 and marked in place *p*2. It may be obvious in this scenario that in order for *p*2 to receive a token, place *p*1 must have received a token first and therefore be in a failed state. However, in the case of Fig. 24, it becomes less obvious when place *p*3 is in a failed state which means failure p1 or p2 caused the transition to enable and fire. To address this issue, the place method will include a Boolean indication for failed state once a transition is enabled, and will not accept another token as a failed state cannot further fail.

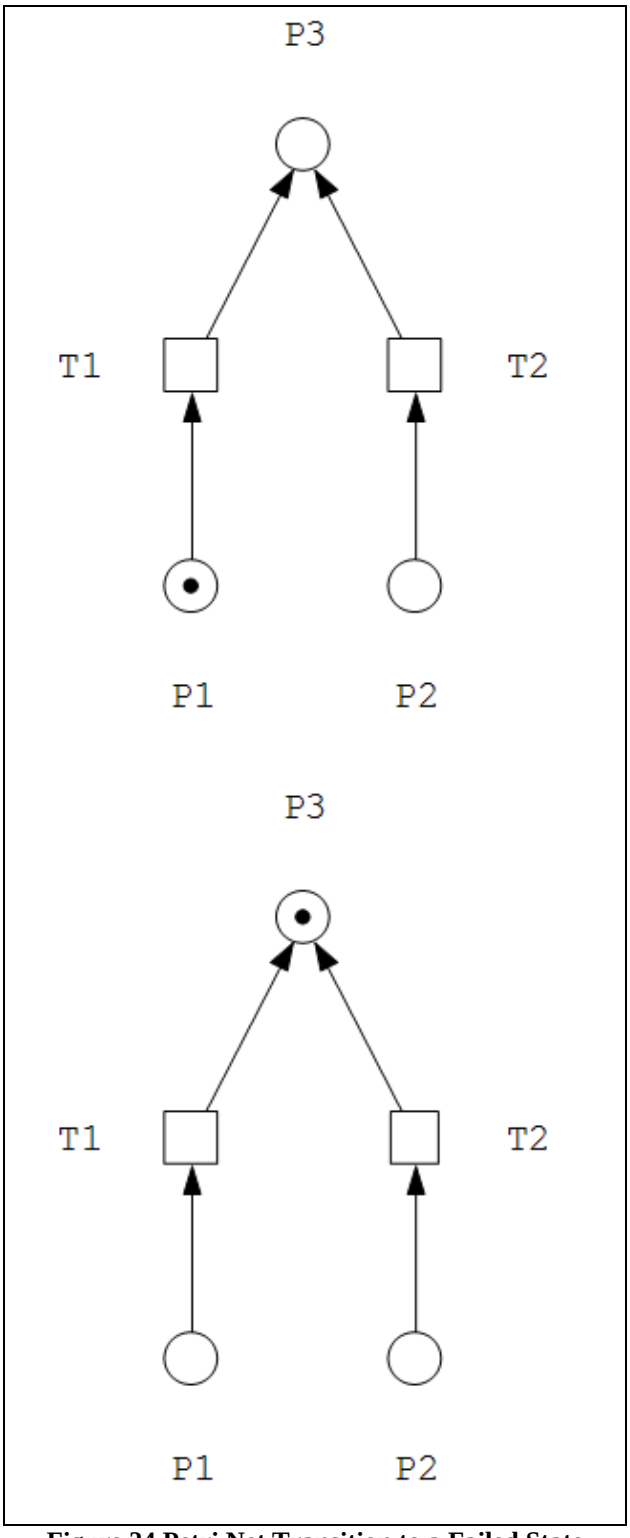


Figure 24 Petri Net Transition to a Failed State

The place, arc and transition classes were developed to represent the modified structure of the traditional Petri net, as highlighted below; see appendix for detail source code.

```
/************************
Place Class
************************/
import java.awt.Point;

public class Place {
  protected int tokens ;
  protected int bound ;
  protected Boolean bound_state;
  protected String name ;
    protected Point coord;

  public Place() {
    this(0) ;
  }

  public Place(int t, int b, b   odelinc) {
    tokens = t ;
    bound = b ;
    bound_state = c;
    coord = new Point();
  }
}
/************************
Arc Class
************************/
class Arc {
  public Place place ;
  public int weight ;
  public int bound ;

  public Arc(Place p) {
    this(p,1) ;
  }

  public Arc(Place p, int w) {
    place = p ;
    weight = w ;
    bound = p.bound ;
  }
}
/************************
Transition Class
************************/
import java.util.HashSet ;
import java.awt.Point;
```

```
public abstract class Transition {
  protected HashSet in, out ;
  protected Boolean immediate ;
  protected String name ;
    protected Point coord;

    public abstract void onFire() ;

  public void addIn(Place p) {
    addIn(p,1) ;
  }

  public void addIn(Place p, int weight) {
    synchronized(in) {
      in.add(new Arc(p,weight)) ;
    }
  }

  public void addOut(Place p) {
    addOut(p,1) ;
  }

  public void addOut(Place p, int weight) {
    synchronized(out) {
      out.add(new Arc(p,weight)) ;
    }
  }

  public void setImmediate(boolean imm) {
    immediate = imm ;
  }
}
```

The place class has been coded to include another parameter for its place called bound_state, which is a Boolean variable to indicate whether or not a transition has been enabled based on that state. If the place occupied a token and enabled a transition, this indicates a failed state and therefore the bound_state would result in a true. In Fig. 25, place *p*1, which contained a token that enabled transition *t*1 to fire, was marked as true for the bound_state, therefore a visual indication will be present to signal to the user that place *p*1 has failed instead of place *p*2.

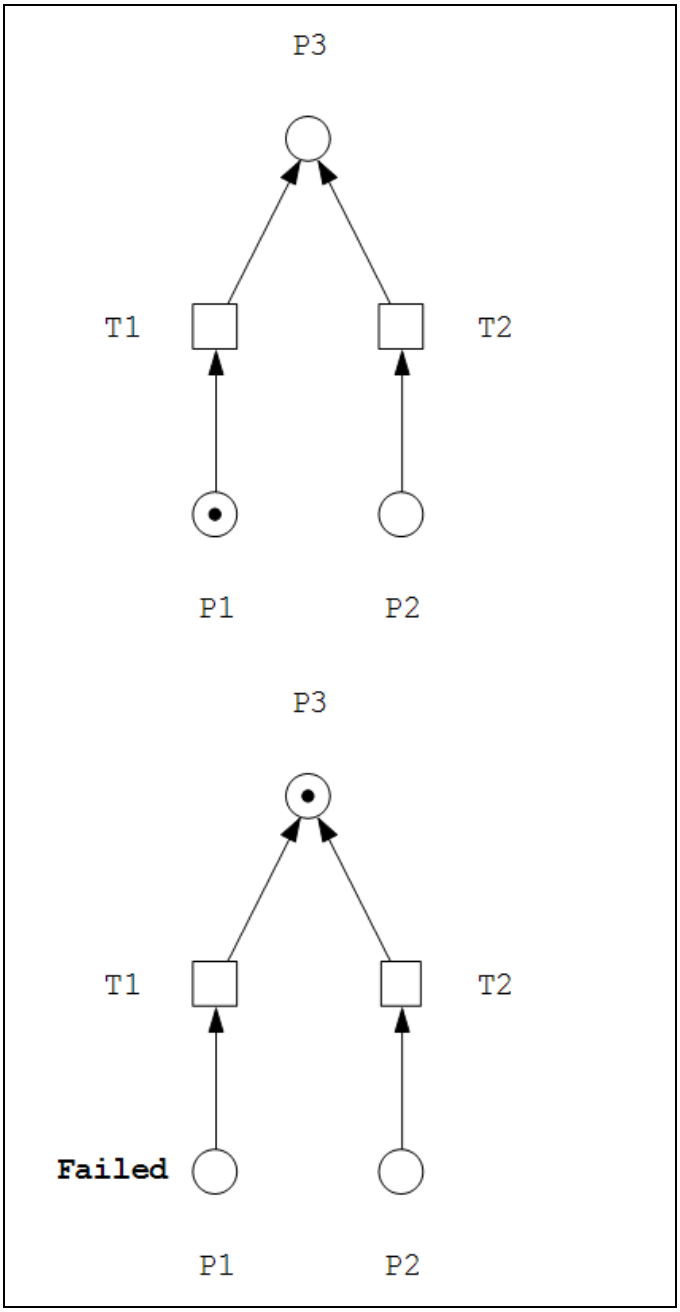


Figure 25 Petri Net Failed State with Failed States

Petri net users can also use arc weights to distinguish between various events when setting up their model using the simulator. Arc weights default at 1, which is represented by a black dot or token. Using Petri net simulations, users can modify the arc weight of transitions as a visual indicator of failure rates or probability of failure. Visually, this can be achieved by

increasing the arc weight for basic events with lower rates of failure and use the default for basic events that exhibit average failure rates. If the average failure rate for the basic event in the system is $10^{-4}hr^{-1}$, and the lower failure rates were in the $10^{-5}hr^{-1}$ range, this can be represented in the simulator by a higher arc weight for the lower failure rate. Increasing arc weight from the default value offers many advantages for the simulator's GUI as this provides a visual indication which distinguishes basic events, which are more prone to failure, to ones with a longer life span which do not require the same surveillance.

3.7.2 Petri Net Simulation

The second limitation this thesis will address in traditional Petri net software is the absence of the feature to generate initial markings without user input. This feature is beneficial for simulation purposes as it creates a sense of randomness and may produce scenarios the user had not predicted. This feature will assist with the reachability concept simulation to determine if an initial marking M_0 will produce a system failure at M_k, where k is the top event.

Traditionally, the transition function is written to require an in and out place and an associated arc weight as requisites to enable the transition. Once the in place consists of enough tokens to meet the requirements of the arc weight, the transition enables and fires the token to the out place. This thesis introduces a new type of transition function which requires no in place or arc weight requisite to become enabled. This initial transition function can have an immediate fire time or be user conditioned to fire; therefore these transitions will be enabled at time 0 and will create the initial markings M_0.

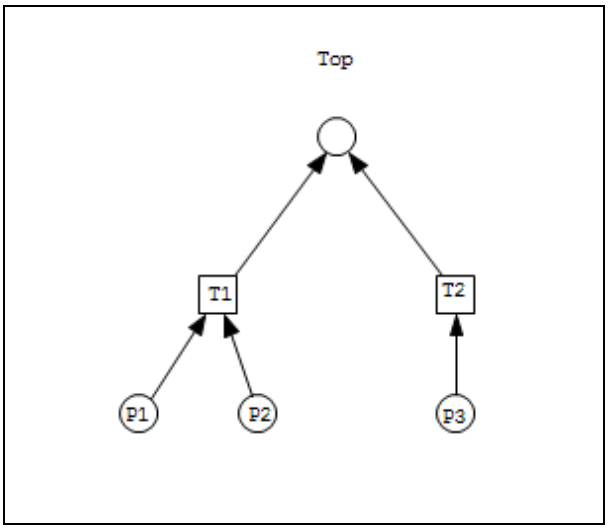


Figure 26 Traditional Petri Net

Figure 26 represents a typical Petri net model for system failure with the top event at the top represented as a place with basic events at the bottom: P1, P2 and P3. Transitions T1 and T2 represent the logic gates AND and OR respectively. There are no initial markings in Fig. 26, thus when we use the simulation function, nothing will happen because there are no tokens in places P1, P2, or P3, therefore the transition cannot meet the requirements to become enabled and has nothing to fire.

The original transition class has been modified to allow transitions to enable without an in place by setting the weight arc to be zero. This way the transitions are constantly enabled and ready to fire.

```
import java.util.HashSet ;
import java.awt.Point;

public abstract class initialTransition {
  protected HashSet  out ;
  protected Boolean immediate ;
  protected String name ;

  public initialTransition () {
    immediate = true ;
    out = new HashSet() ;
  }
```

```
  public abstract void onFire() ;

 public void addIn(Place p) {
  addIn(p,0) ;
 }

 public void addIn(Place p, int weight) {
  synchronized(in) {
   in.add(new Arc(p,0)) ;
  }
 }

 public void addOut(Place p) {
  addOut(p,1) ;
 }

 public void addOut(Place p, int weight) {
  synchronized(out) {
   out.add(new Arc(p,weight)) ;
  }
 }

 public void setImmediate(boolean imm) {
  immediate = imm ;
 }

 public void setName(String name) {
  this.name = name ;
 }

 public String getName() {
  return name;
 }
}
```

If the input to the transition is null, the transition will be true and enable. This represents a case where the user is trying to simulate a system with no initial markings and the transitions fire without user control.

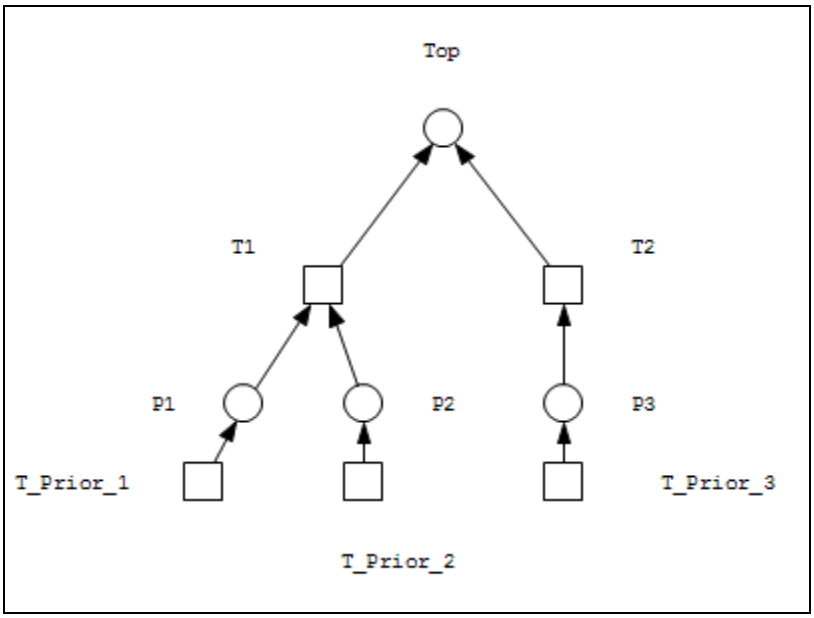


Figure 27 Modified Petri Net

The user would set up Fig. 27 such that the transitions T_Prior have no inbound places and only an out bound place with a default weighted arc so that they are enabled at time t_0. The initial transitions are linked to the basic event places P1, P2 and P3, while the transitions T1 and T2 are linked to their input places, and outbound to the top event place. The code to model the Petri net in Fig. 27 is shown below:

```
import java.awt.*;
import java.io.*;

public class PetriTest extends PetriNet {
   public static void main(String[] args) {
        PetriTest test = new PetriTest();

        Place p1 = new Place(0);
        p1.setName("P1");

        Place p2 = new Place(0);
        p2.setName("P2");

        Place p3 = new Place(0);
        p3.setName("P3");

        Place p4 = new Place(0);
        P4.setName("TOP");
```

```
        test.add(p1);
        test.add(p2);
        test.add(p3);
        test.add(p4);

        Transition t_prior_1 = new initialTransition();
        t1.setName("T_Prior_1") ;
        t1.addOut(P1, 1) ;

        Transition t_prior_2 = new initialTransition();
        t1.setName("T_Prior_2"” ;
        t1.addOut(P2, 1) ;;

        Transition t_prior_3 = new initialTransition();
        t1.setName("T_Prior_3") ;
        t1.addOut(P3, 1) ;

        Transition t1 = new initialTransition();
        t1.setName("T1") ;
        t1.addOut(P4, 1) ;

        Transition t2 = new initialTransition();
        t2.setName("T2") ;
        t2.addOut(P4, 1) ;

        test.add(t_prior_1);
        test.add(t_prior_2);
        test.add(t_prior_3);
        test.add(t1);
        test.add(t2);
        test.init();
}
```

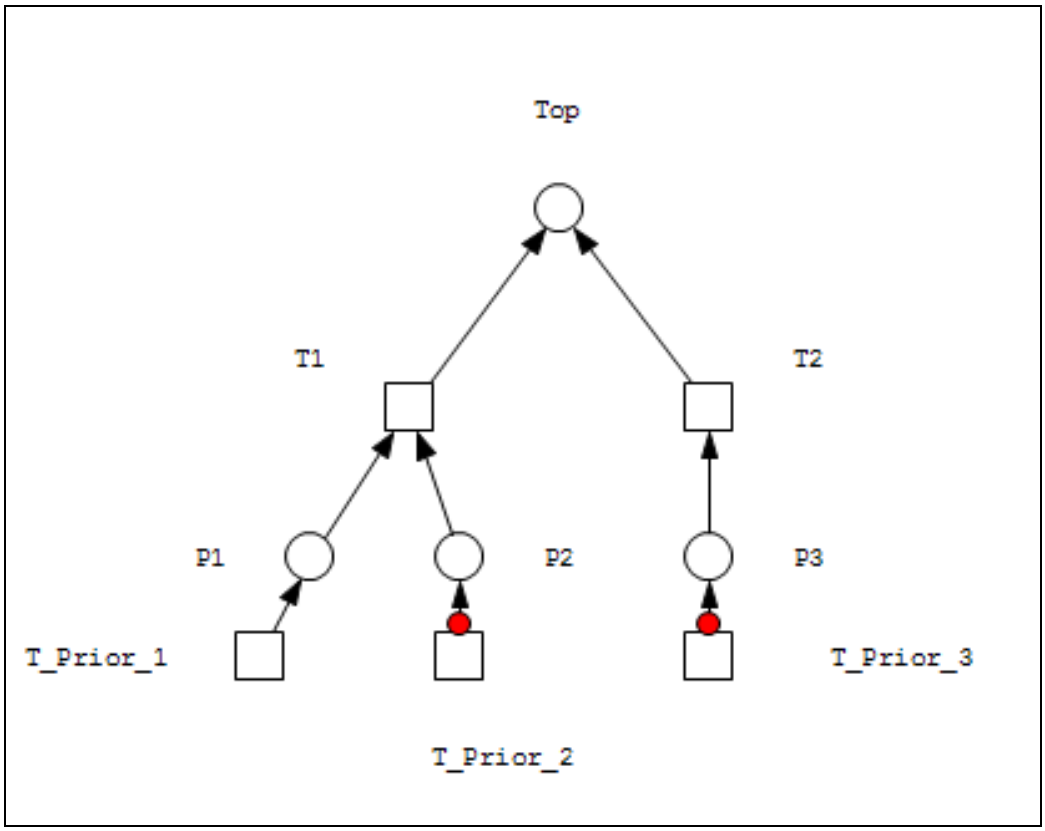


Figure 28 Simulation Markings

In Fig. 27, if the user incorporates initial transitions before the basic events places P1, P2 or P3, and simulate the model, the simulator will populate the basic event places as seen in Fig. 28. This allows the Petri net to be simulated without any user initial markings. The simulation will then proceed the same way as if the user had inputted the initial markings and stop once the top event is populated.

Figure 29 illustrates the issue with simulator generated markings. The existing software package is built based on the original Petri net definition and does not recognize, for the purposes of PRA, that the events will represent the failure of equipment or failed states. There are instances where markings will be fired into already failed states as seen in Fig. 29. This presents a case where a state has already failed, but the traditional Petri net software structure

does not recognize the system failure model and continues to enable transitions which do not accurately reflect the system's behaviour.

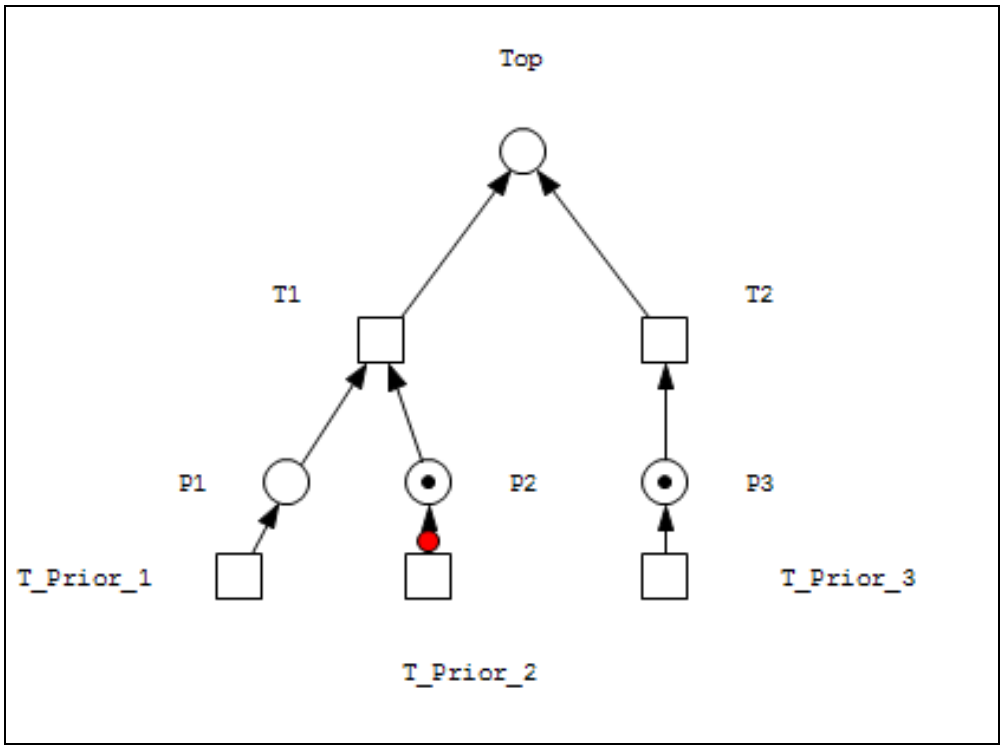


Figure 29 Double Markings

This limitation can be addressed by combining the modified place class from the previous sub section and incorporate the bound_state as an indication of a failed state. If the output to the transition is bound_state true, the transitions will hold off on the firing even if the fire conditions are all true and enabled.

3.8 Chapter Summary

This chapter provides an in-depth assessment of traditional Petri nets and identifies the gaps preventing it from becoming a modeling technique for system failures within PRA. Techniques are developed to revise and tailor traditional Petri net modeling for system failure analysis applications. Methodologies for Petri net constructions for PRA are established to provide a structured approach to perform qualitative and quantitative analysis. Petri net

construction techniques such as building logic gates and transforming existing fault trees to Petri nets are explored. Petri net methods for obtaining initiating events are discussed. Qualitative analysis for obtaining minimal cut and path sets are demonstrated through examples. The fundamentals of quantitative analysis for system unavailability are discussed. Non-coherent systems are introduced, modeling them with Petri nets were shown, and methods for obtaining prime implicants were explored. Lastly, a Petri net simulation software package is developed in expanded detail to overcome the limitations of traditional software programs: such as the absence of non-user generated initial markings, and the inability to model a failed state. The ability to model a failed state is an important characteristic for system failure modeling for PRA. The software classes developed tailored the traditional Petri net software to accurately model and simulate system failures for PRA. The following chapter demonstrates the Petri net as a modeling tool for PRA by exploring its application in an airlock system of a containment boundary, and a Maintenance Cooling System for a design basis accident. These case studies will demonstrate Petri net's ability to model system failure and provide a structured approach for qualitative and quantitative analysis.

CHAPTER 4 APPLICATION OF PETRI NETS IN A CANDU REACTOR

This chapter demonstrates the applications of Petri net for PRA by exploring the airlock system and the maintenance cooling system in a CANDU nuclear power plant. The first case study describes the airlock systems, its operation and equipment used across the plant as per the Defense-in-Depth methodology. A Petri net is used to model system failure and analysis is performed to determine minimal cutsets and system unavailability. The second case study explores the operational side of a CANDU nuclear power plant by studying the maintenance cooling system design function and effects of key equipment failure such as their pumps and isolations valves. The Petri net is constructed as a non-coherent system and the prime implicants are obtained. Both cases establish the importance of design and its effect on future probabilistic analysis.

4.1 CANDU Airlock System for Containment Boundary

The top objective for any nuclear power plant is to safely produce power. During any design basis accident, a nuclear power plant must prevent or limit the release of radioactive material to the public and environment. In a CANDU reactor, the reactor is enclosed by a containment building built from concrete. The only access to and from the containment building are the airlock doors located on various levels of the building.

An airlock is a chamber in the containment wall of the reactor vault with two doors. One door opens to the inside of the reactor vault, and the other door to the outside of the reactor vault.

The airlock, which is considered part of the Negative Pressure Containment (NPC) system, is required to perform the safety function of maintaining the containment boundary. At least one airlock door must be closed with sufficient pressure maintained in the seal to ensure preservation of the containment boundary. This implies that both sets of seals on each door are qualified. Airlock door seals may be deflated only when the door is in use. All airlocks and transfer chambers have valves to equalize the pressure across the service and containment doors before the doors are operated.

Following all DBA conditions, including all LOCAs (Large, Small, In-Core), Flooding Events (FE) (other than LOCA or Feedwater), Fuel Handling System Failures (FHSF), LOCA+ Loss of Emergency Coolant Injection (LOECI) and Secondary Side Line Break (SSLB) events, the credited safety function of the airlock door seal is to remain inflated with sufficient seal pressure to ensure a containment boundary. Electromechanical indicators on the containment panel in the control room are operated by pressure switches monitoring the pressure in the air seals and by limit switches monitoring the state of the doors. The components associated with the door seals must maintain a pressure boundary and not leak externally for up to three months.

Maintaining pressure boundary is crucial to remaining compliant with the safety objectives in place for nuclear power plants. A potential leak or equipment failure at this point would prove costly.

4.2 Case Study I: Airlock Safety System during a Design Basis Accident

In this case, the scenario involves a DBA occurrence and the instrument air system is now offline. The airlock seals, which have a credited safety function to maintain pressure boundary for up to three months, are switched to back up air supply tanks. For a simplified airlock system, the top event is failure to maintain pressure boundary. The causes for this failure are failure of the equalizer valve, door fails to close or hatch is not locked, and the inflatable seals fail to perform their credited safety feature.

Equalizer valves are designed to equalize the pressure between the reactor bay and service side. They function by opening vents in the equalizer valve to allow airflow to enter the airlock chamber until pressure is equalized, and then the vents in the equalizer valve will close. The equalizer valve vent area is designed to limit the ingress of air into the containment to minimize the emergency filtered air discharge system capacity. A set of pipes lead to the service side, and another set to the reactor bay. The equalizer valves fail when the gear box fails, which prevent the vents from opening or closing to allow constant flow between the reactor bay and service side. The equalizer valves' safety function cannot be met when the exhaust pipe cannot close or seal properly.

To meet the safety function for maintaining pressure boundary, the airlock doors must be closed by a latch or else a beacon light and annunciation to the MCR will occur. If the door is not properly closed and latched, the equalizer valves and seals would not be available to provide their safety functions either. The instrumentation provided for airlocks are required to control pneumatically operated doors, and provide indication of low air pressure in inflatable door seals and indication of the state of doors (open/closed).

In the event of a DBA, the airlock door seals are designed to inflate thus creating the necessary seal to maintain pressure boundary. During normal operation, the door seals are inflated via the instrument air system. During a DBA however, the inflation of the seals are switched to the back-up air supply tank. This presents a potential for system failure because of the limited air supply compared to during normal operation when a constant supply came from the instrument air system.

FMEA is a design tool used in reliability assessment and is recognized as an essential function in design. It is defined as "a systematic process for identifying potential design and process failures before they occur, with the intent to eliminate them or minimize the risk associated with them" [IMCA, 2002]. The purposes of an FMEA are as follows [IEEE Std.352, 1987]:

1. To assist in selecting design alternatives with high reliability and high safety potential during early design phases.
2. To ensure that all conceivable failure modes and their effects on the operational success of the system have been considered.
3. To list potential failures and identify the magnitude of their effects.
4. To develop early criteria for test planning and the design of test and checkout systems.
5. To provide a basis for quantitative reliability and availability analyses.
6. To assist in the objective evaluation of design requirements related to redundancy, failure detection systems, fail-safe characteristics, and automatic and manual override.

Table 7 demonstrates the FMEA for the airlock system and describes the function for each component, modes of failure which affect the top event, mechanisms which can fail, and their overall effect on the system. Based on the behaviour properties of the

equipment identified in Table 7, their associated failure rates were identified using data from IEEE and IAEA as seen in Table 8. The data obtained in Table 8 will be used for quantitative analysis of the overall airlock system to determine the failure probability of the top event – failure to maintain pressure boundary.

Table 8 FMEA of the Airlock System

Failure Mode and Effects Analysis – Airlock System				
Component	Function	Failure Mode	Failure Mechanism	Effect on System
Equalizer valve	Ensure equalization	Failure to Function	Gear box fails Exhaust pipe fails to close	Airlock system fails to maintain pressure boundary
Seals	Inflate on demand Deflate on demand	Leakage Rupture	Piping System Failure	Airlock system fails to maintain pressure boundary
Door	Locks on demand Opens on demand	Fail Closed Fail Open	Locking mechanism failures	Airlock system fails to maintain pressure boundary
Tank	Engages on demand	Fail Open Leakage	Does not engage No air in tank	Seals fail per design function
Valve	Operates on demand	Fail Closed Fail Open Failure to function	Fails per design function	Seal fail per design function

Table 9 Airlock Equipment Failure Rates

Equipment	Failure Type	Failure Probability (Failures/10^6 Cycles)	Failure Rate (λ-Hr)	Source
G1 – Gearbox	RATE	0.32E-06	0.3E-06	[IEEE Std. 352, 1987] [IAEA-TECDOC-930, 1997]
E1 – Exhaust pipe	RATE	0.34E-06	0.3E-06	[IEEE Std. 352, 1987] [IAEA-TECDOC-930, 1997]
D1 – Door	RATE	0.50E-06	4.5E-06	[IEEE Std. 352, 1987] [IAEA-TECDOC-930, 1997]
S1 –Seal	RATE	1.34E-06	0.42E-06	[IEEE Std. 352, 1987]
V1 – Valve	RATE	1.62E-06	0.89E-06	[IEEE Std. 352, 1987]
P1 –Piping system (small)	RATE		2.4E-06	[IAEA-TECDOC-930, 1997]
P2 –Piping system (Big)	RATE		1.4E-06	[IAEA-TECDOC-930, 1997]
T1 – Tank	RATE		1.21E-06	[IEEE Std. 352, 1987]
T2 – Back up tank fails to engage	RATE	63.2E-06	0.023E-06	[IEEE Std. 352, 1987]

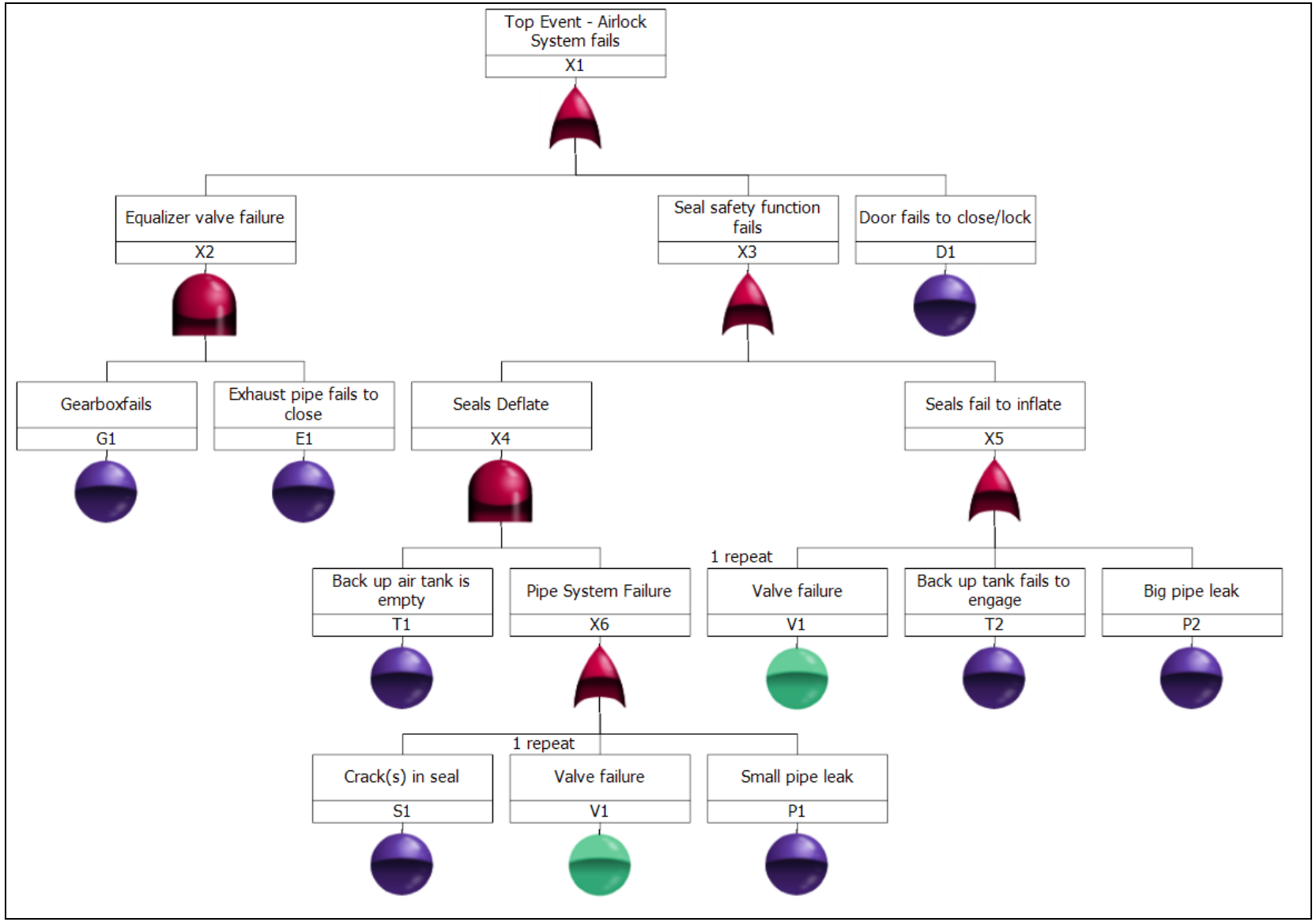


Figure 30 Airlock System Fault Tree

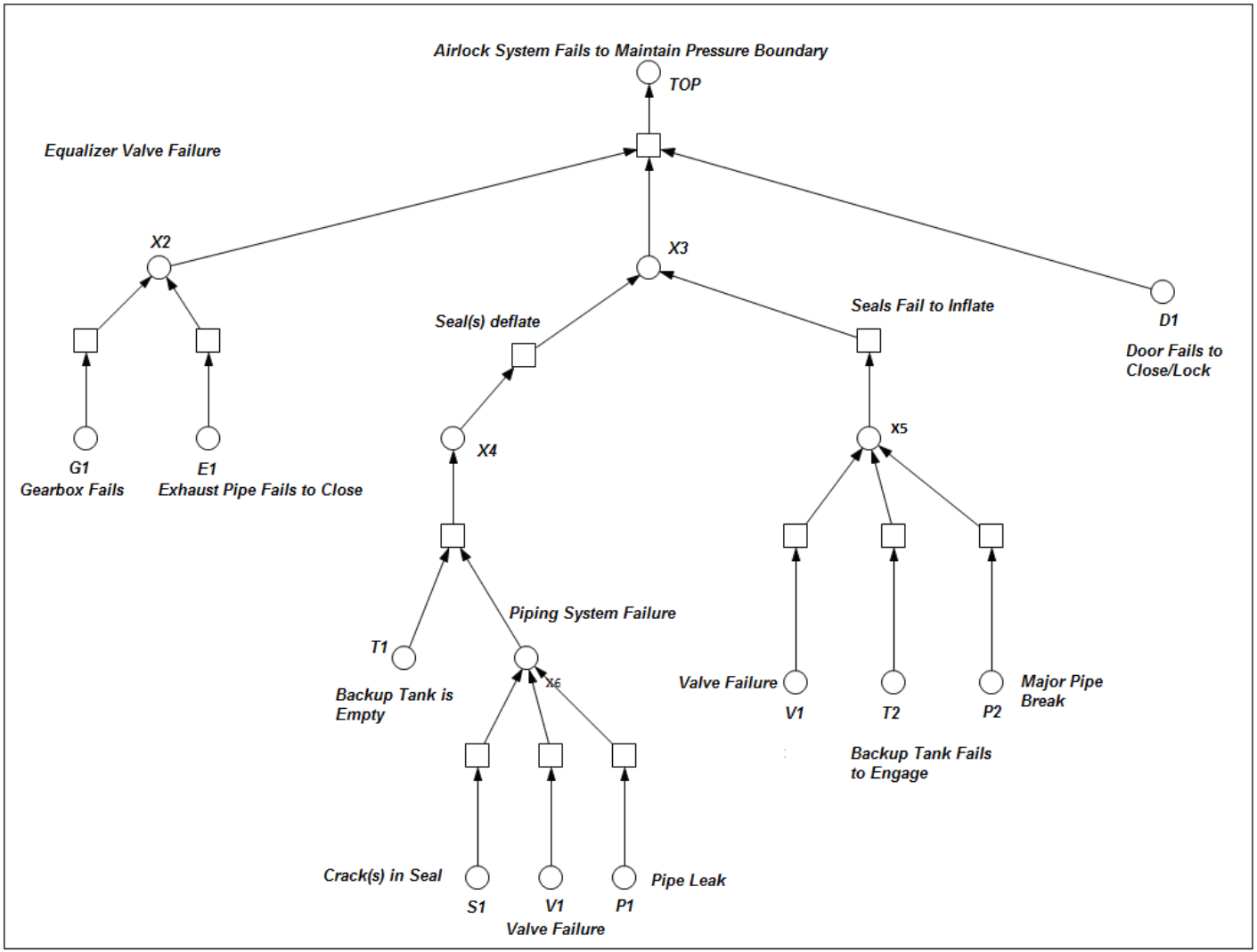


Figure 31 Airlocks System Petri Net

The contributing factors to airlock system failures include:

- Seals deflating during the course of the accredited safety function
- Back up tank being empty
- Cracks in the seal leading to loss of air
- Pipe leak leading to loss of air (causing deflation)
- Valve failure, including check valves, allowing back flow or relief valves to release prematurely
- Seals do not inflate after a Design Basis Accident
- Back up tanks do not engage
- Major pipe leak (preventing inflation)
- Valve failure, including check valves, not allowing positive flow and relief valves to constantly vent

Figure 30 and 31 models the airlock system in fault tree and Petri net format respectively. The fault tree model is used to verify the qualitative analysis of the Petri net model. The variables represent the following failure events:

- G1 – Gearbox fails
- E1 – Exhaust pipe does not close or seat
- D1 – Door fails to close or lock
- S1 – Crack(s) in seal
- V1 – Valve failure; check valve preventing flow or allowing back flow, relief valve prematurely releasing or not closing, any valve that operates outside of the design function causing leaks

- P1 – Small Pipe leaks in the piping system (piping small, <= 2.5cm diameter, 169 welds)
- P2 – Medium pipe leaks in the piping system (piping medium, 1"< diameter)
- T1 – Back up tank is empty
- T2 – Back up tank fails to engage

Table 10 Minimum Cutset For Airlock System Based On The Petri Net Model

Minimum Cutset for Figure 31		
Step 1	X1	Replace X1
Step 2	X2 X3 D1	Replace X2
	G1, E1 X3 D1	Replace X3
	G1, E1 X4 X5 D2	Replace X4
	G1, E1 T1, X6 X5 D1	Replace X6

Table 11 Minimum Cutset For Airlock System Based On The Petri Net Model (Cont'd)

	G1, E1 T1, S1 T1, V1 T1, P1 X5	Replace X5
Step 3	G1, E1 T1, S1 T1, V1 T1, P1 V1 T2 P2 D1	

The qualitative analysis is performed first based on the Petri net model. The details are shown in Table 9. A simplified airlock system was modeled for system failure, and based on the number of basic events only two orders of cutsets were generated.

The minimum cutsets produced by the Petri net model are:

1st Order: [D1], [P2], [T2], and [V1]

2nd Order: [G1, E1], [T1, P1], and [T1, S1]

To verify the qualitative analysis of the Petri net model, a fault tree model is constructed using Relex, and minimal cutsets are obtained as shown in Fig. 32. The results of the Relex calculation for minimum cutsets yield the same results as the Petri net model.

Using the method discussed in chapter 3, the minimal path sets for Fig. 31 are also solved, as shown in Table 10.

Table 12 Minimum Path Set For Airlock System Using The Petri Net Model

Minimal Path set for Figure 31		
Step 1	X1	Replace X1
Step 2	X2, X3, D1	Replace X2
	G1, X3, D1 E1, X3, D1	Replace X3
	G1, X4, X5, D1 E1, X4, X5, D1	Replace X4
	G1, T1, X5, D1 G1, X6, X5, D1 E1, T1, X5, D1 E1, X6, X5, D1	Replace X6
	G1, T1, X5, D1 G1, S1, V1, P1, X5, D1 E1, T1, X5, D1 E1, S1, V1, P1, X5, D1	Replace X5

Table 13 Minimum Path Set For Airlock System Using The Petri Net Model (Cont'd)

Step 3	G1, T1, V1, T2, P2, D1 G1, S1, V1, P1, V1, T2, P2, D1 E1, T1, V1, T2, P2, D1 E1, S1, V1, P1, V1, T2, P2, D1	Reduce
Step 4	G1, T1, V1, T2, P2, D1 G1, S1, V1, P1, T2, P2, D1 E1, T1, V1, T2, P2, D1 E1, S1, V1, P1, T2, P2, D1	

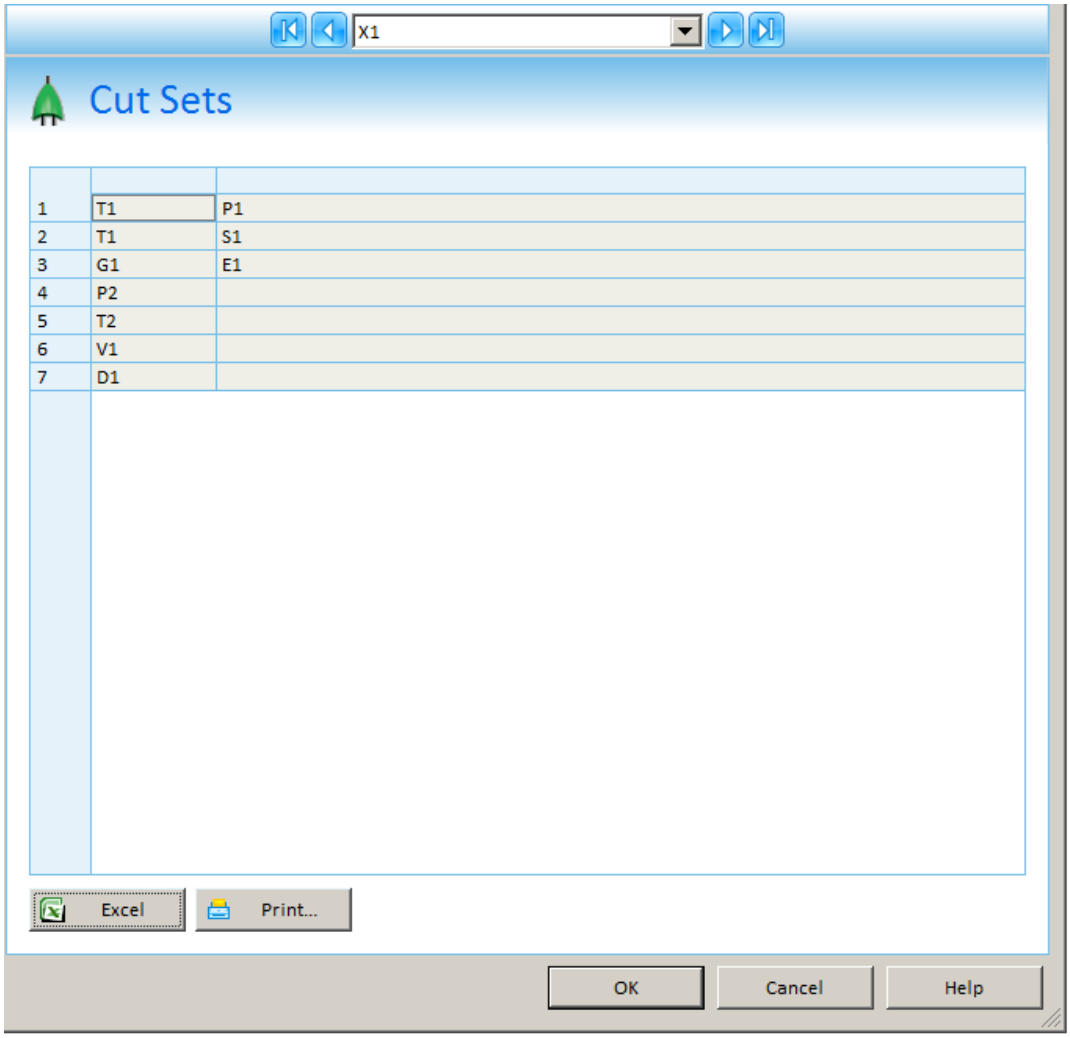


Figure 32 Relex Qualitative Analysis

From the results obtained from the qualitative analysis, quantitative analysis can be performed to calculate the system unavailability Q_s (t). From chapter 3 Eqn (3.10), system unavailability can be calculated by determining the unavailability for 1st and 2nd order cutsets.

1st order unavailability calculation:

$Q_s(t) = 1-[1-Pr\{P1\}][1-Pr\{P2\}]\ldots[1-Pr\{P_n\}]$

where,

P1 = Pr{D1}, P2 = Pr{P2}, P3 = Pr{T2} and P4 = Pr{V1}

$Q_s(t) = 1-[1-Pr\{D1\}][1-Pr\{P2\}][1-Pr\{T2\}][1-Pr\{V1\}] = 6.813E-09$

2nd order unavailability calculation:

where,

P1 = Pr{G1}Pr{E1}, P2 = Pr{T1}Pr{P1} and P3 = Pr{T1}Pr{S1}

$Q_s(t) = 1-[1-Pr\{G1\}Pr\{E1\}][1- Pr\{T1\}Pr\{P1\}\ ][1- Pr\{T1\}Pr\{S1\}] \approx 0$

Based on the failure rates, the second order unavailability is approximately zero and will not affect the overall unavailability of the top event. Third order sets are not required for this calculation because the occurrence probability would be even less likely and thus would not affect the top event system unavailability. Therefore, the system unavailability of the top event occurrence is 6.81E-09. This result is verified by performing the calculation using the same failure rates for the constructed fault tree in Fig. 30. It can be shown that these results are consistent. The Relex results are shown in Fig. 33 and Fig. 34 and Table 11 summarizes the above results.

Table 14 Airlock System Quantitative Analysis Summary

Quantitative analysis for Figure 31		
1st Order sets	[D1] = 4.50E-09 [P2] = 1.40E-09 [T2] = 2.30E-11 [V1] = 8.90E-10	Calculation: Qs(t) = 1-[1-Pr{D1}][1-Pr{P2}][1-Pr{T2}][1-Pr{V1}] = 6.813E-09
2nd Order sets	[G1, E1] = G1*E1 = 9.00E-20 [T1, P1] = T1*P1 = 2.90E-18 [T1, S1] = T1*S1 = 5.08E-19	Calculation: Qs(t) = 1-[1-Pr{G1}Pr{E1}][1-Pr{T1}Pr{P1}][1-Pr{T1}Pr{S1}]
Top Event X1		6.81E-09

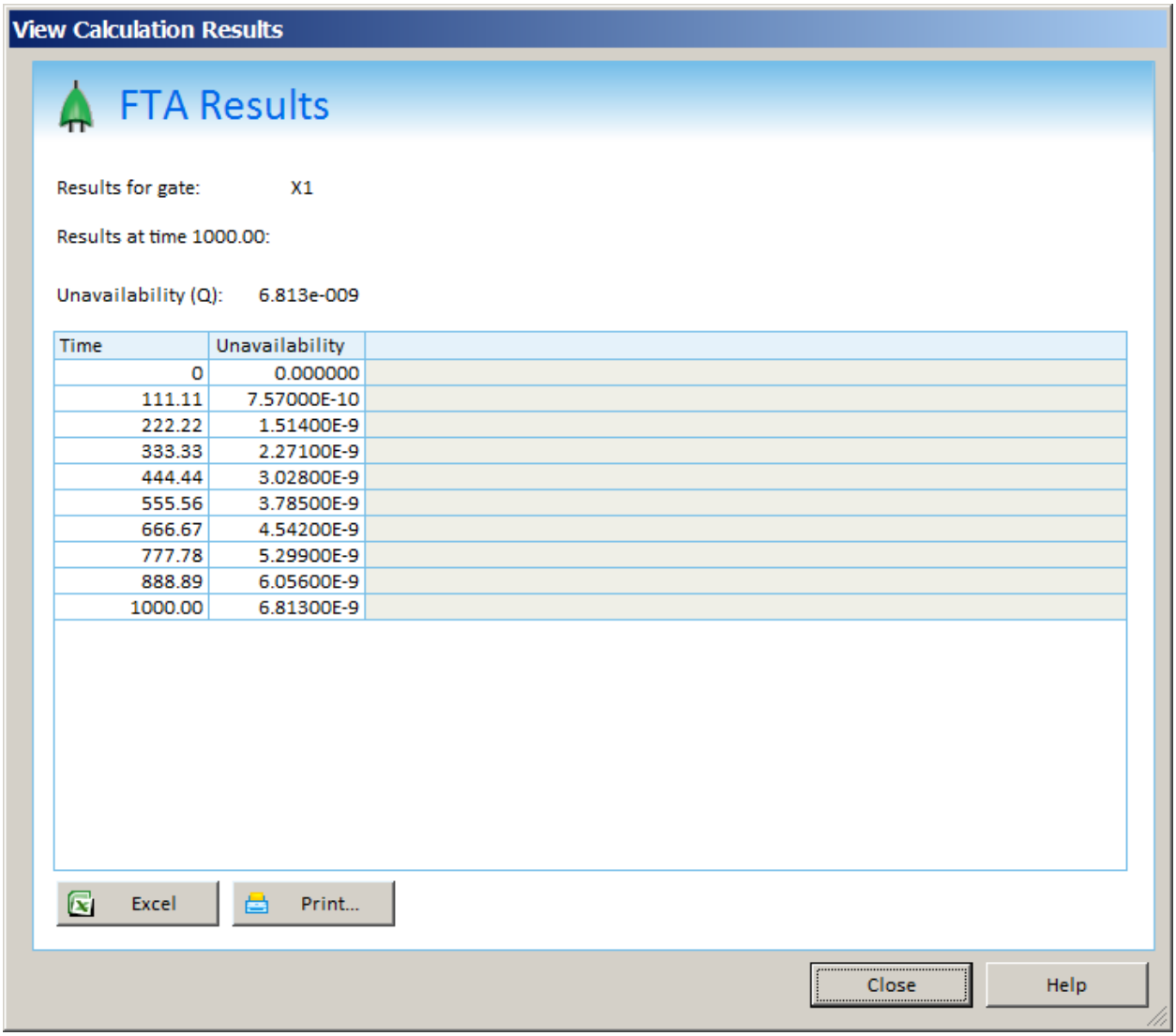


Figure 33 Results of Quantitative Analysis

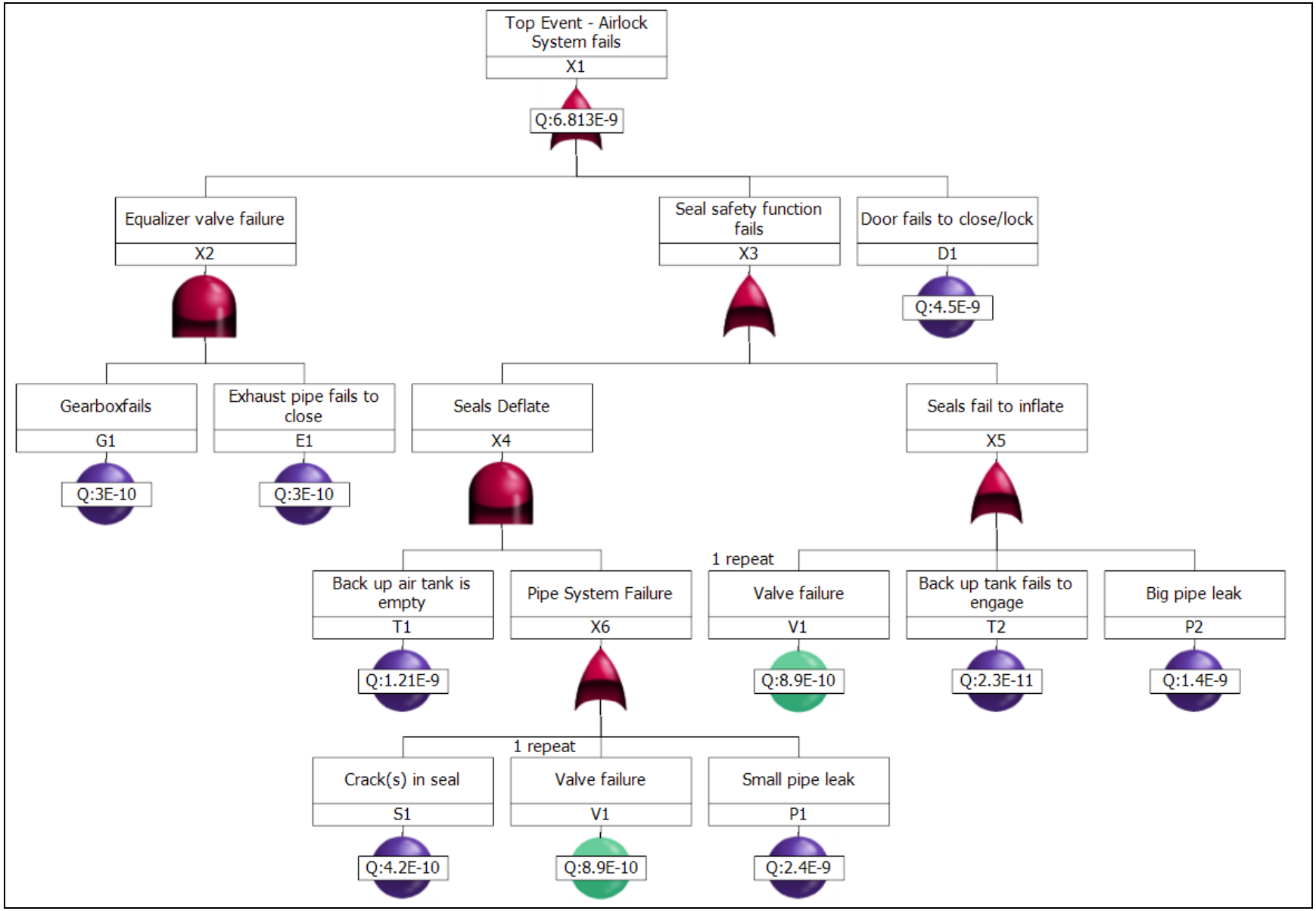


Figure 34 Relex Quantitative Analysis

4.3 Case Study I: Results and Discussion

Case study 1 explores the containment boundary system failure for an airlock system. The basic events for the system failure are identified. The fault tree and the Petri net showing the logical connection of the basic events and the top event, failure to maintain pressure boundary, are constructed. The Petri net is modeled in a top-down orientation to utilize the tools in chapter 3 to perform qualitative analysis. Both the fault tree and the Petri net produce the same results with respect to qualitative and quantitative analysis. The qualitative analysis identifies two sets of minimal cutsets, the first order and second order. The first order cutsets represent a single basic event, which, if to occur, would cause the top event to occur. These basic events include the failure of equipment leading to the seals' failure to inflate (V1, T2, and P2) and the door's failure to lock (D1), which compromises and prevents the airlock system to perform its design function. The second order cutsets represent pairs of basic event occurrences that result in system failure. This includes the equipment failure which causes seal deflation through the failure method of leakage. The other combination for this second order cutset includes the failure of the equalizer valve. Once the minimal cutsets have been determined, the quantitative analysis can be used to calculate the system unavailability. Each component of the system is further reviewed using the FMEA, and failure rates are obtained from international databases such as IEEE and IAEA. It is determined that the system unavailability is 6.81E-09, which according to [IEC61508], represents an improbable; very unlikely to occur event. The results of the quantitative analysis are acceptable as the probability of the airlock system failing is minimal. This case study shows that Petri nets are an effective tool which can be used to construct and model systems used for PRA.

4.4 Case Study II: Maintenance Cooling System

The maintenance cooling system, which cools the heat transport system from 177°C to 58.6°C, utilizes separate preheaters for heat rejection. The main heat transport pumps circulate heat transport fluid through the preheaters where heat is removed by the shutdown cooling system, by circulating light water through the shell of the preheaters. In order to operate the main heat transport pumps, it is necessary to keep the heat transport system pressurized. Hence, when the shutdown cooling system is in operation, it is not possible to open the boiler channel cover for tube sheet inspection or tube plugging to service the main heat transport system pump glands or internals, or to carry out maintenance on the emergency injection isolating valves. Therefore maintenance cooling system is used to provide fuel cooling with the heat transport system depressurized and drained to header level.

The failure of both maintenance cooling system pumps will result in loss of coolant flow to the fuel. Both MCS pumps are on Class III power. If Class IV power is lost, cooling to the core will be interrupted for four minutes before Class III power is re-established. This will result in a D_2O temperature increase of about 28°C in the channels before cooling is restored. If one MCS pump fails, cooling will be provided by the second pump.

If the Primary Heat Transport (PHT) system is open for maintenance, the heat transport system D_2O level should be maintained so that the fuel is cooled by pool boiling. The heat transport system should be closed up, filled, and pressurized as soon as possible to assure adequate long term cooling of the fuel if maintenance cooling cannot be restored. Maintenance Cooling System reliability consideration has been given to the

effects on the PHT system if circulation through the core is interrupted on low level operation because of failure in the maintenance cooling system. This is particularly important when the heat transport pump is open for maintenance because the heat transport system is open to the environment.

When the station encounters a DBA, there are no Environmental Qualification (EQ) requirements for MCS and therefore the actuators do not have an active safety function. However, the torque/limit switch compartment contains 48 VDC control circuits. Moisture due to accident conditions may produce a ground fault which could spuriously operate the actuator.

Case study II explores the scenario where an electrical connector that functions per design can have adverse impact to their respective safety systems. System vibration has caused degradation of the control power connector on some applications where Quick Disconnect Couplings (QDC's) were installed for EQ. The system vibration has caused wear on the 14-conductor pin-and-socket connector to a point where control power is lost or intermittent on some logic circuits in the connector. A replacement connector has been proposed and designed, which is a "solid' connector (there are no pin-and-socket connections). This replacement connector will be installed in locations that have been found to be susceptible to the system vibration. The scenario will explore the roles of the D_2O isolation valves within the MCS and how they affect the flow temperature

This simplified Petri net is constructed, as shown in Fig. 35, to represent the output flow temperature of the PHT system during maintenance activities. The MCS is engaged and the major events are D_2O pump and/or isolation valves operate normally as shown in event 2, and pumps and/or isolation valves fail as shown in event 3. The top

event ‘T’ of this model is the case where the outflow temperature is high and therefore MCS fails to sufficiently cool the heat transport system.

It should be noted that in Fig. 35, events 2 and 3 are mutually exclusive; event 2 is a pump/isolation valve success state, and event 3 is a pump/isolation valve failure state.

Table 15 PHT Non-Coherent System

Step 1	T
Step 2	B,3
Step 3	1,3 3’,3
Step 4	[1,3], [3’, 3]

The procedure to determine the path sets of the system is shown in Table 12. The next step is to remove inconsistent path sets from the outputs. In step 4, the only inconsistent path set is [3’, 3] because they are mutually exclusive events. Therefore, it can be seen that the Set {3’, 3} is inconsistent and the only path set is {1, 3}.

Top event non-occurrence T’ can also be expressed as:

$$\overline{T} = \overline{1} \cdot \overline{3}$$

Where the events 1’ and 3’ are heat transport in service and isolation valve normal, respectively. The minimal cutsets can be obtained by taking the complement of the top event T’.

$$\overline{\overline{T}} = T = \overline{\overline{1} \cdot \overline{3}} = \overline{\overline{1}} + \overline{\overline{3}} = 1 + 3$$

Therefore the prime implicants of Fig. 35 are [1] and [3], which means during PHT maintenance the failure of the MCS pumps/isolation valves would result in the inability to provide adequate cooling.

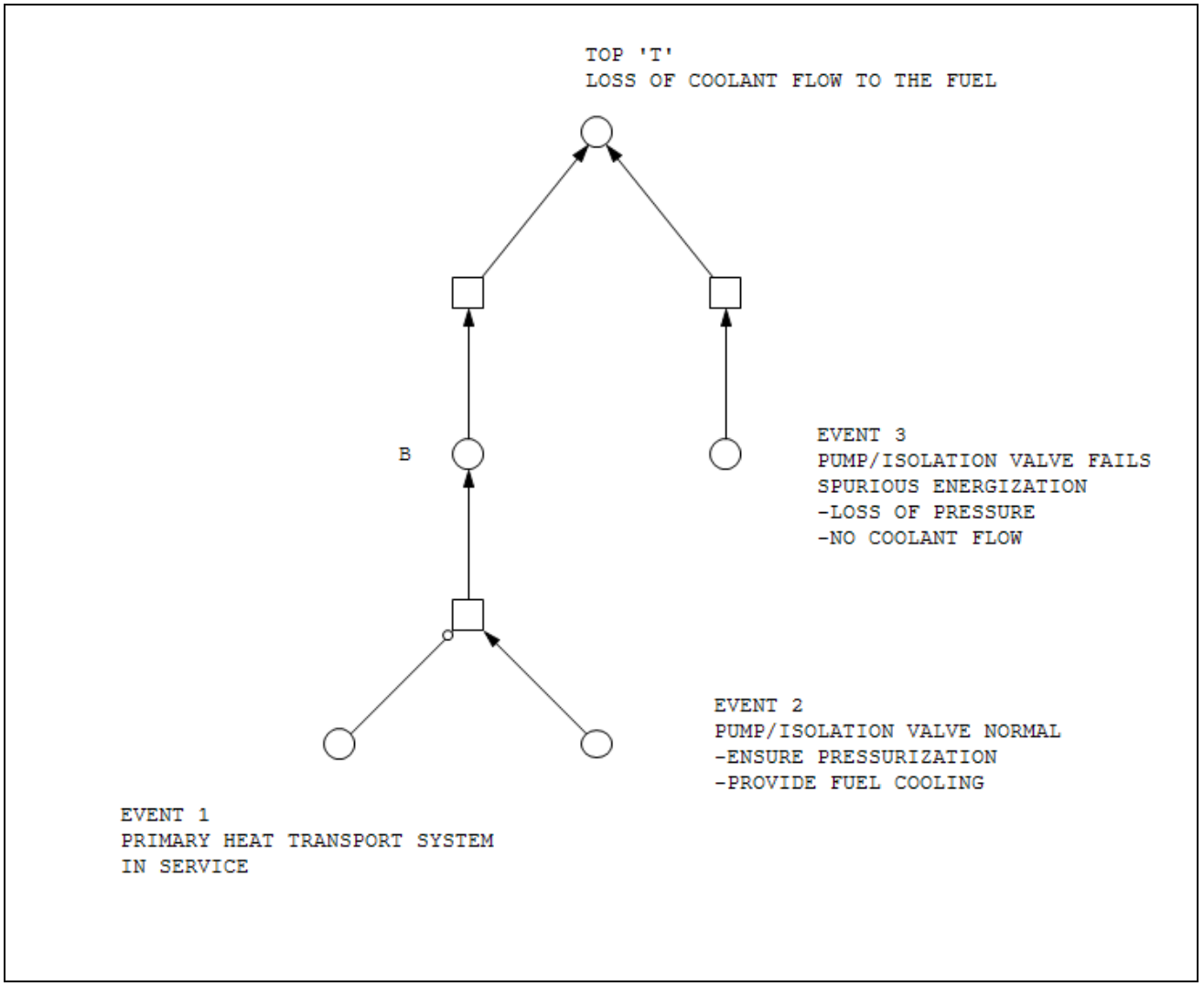


Figure 35 Maintenance Cooling Petri Net

4.5 Case Study II: Results and Discussion

Case study II explores the maintenance cooling system and discusses maintenance and outage station window tasks dealing with the PHT. The Petri net model contains a mutually exclusive binary event, and using the same approach described in chapter 3 to find minimum cutsets, does not always produce the correct results. When the top down methodology is applied to Fig. 35, the cutsets produced will be {1, 2} and {3}. This result is not completely accurate because the minimum cutset {1} cannot be obtained. The minimum cutset {1} is clearly correct to the analyzer and for practical applications and sets {1} and {1, 2} are essentially the same. This case study demonstrates the capability of Petri nets to model non-coherent systems. Qualitative analysis is performed to yield the prime implicants of {1} and {3}.

4.6 Chapter Summary

This chapter demonstrates the applications of Petri net as a modeling tool for PRA for the airlock system and the MCS in a CANDU nuclear power plant. During a DBA, the airlock system has an accredited safety function to maintain pressure boundary. The airlock system is constructed and modeled using Petri nets. Qualitative analysis is performed based on the model and the results are compared to those based on fault tree analysis. The data for equipment failure was obtained from IEEE and IAEA and quantitative analysis for system unavailability were performed then verified using traditional methods. Both coherent and non-coherent cases were explored to demonstrate the ability of Petri nets to graphically model systems and to provide a structured approach for qualitative and quantitative analysis. The two cases have demonstrated that PRA modeling can be performed by Petri nets while maintaining the techniques available to perform qualitative and quantitative analysis.

CHAPTER 5 CONCLUDING REMARKS AND RECOMMENDATIONS FOR FUTURE RESEARCH

5.1 Concluding Remarks

A brief history of the need for safety and the chronology of the need for PRA are presented. Petri net theory is introduced as an alternative modeling tool for PRA. The thesis demonstrates Petri net modeling methods for system failure analysis by making use of logic functions. Transformation of existing fault trees into Petri nets is also possible by converting logic gates into their equivalent Petri net sets; this bridges the gap between the two modeling methods. The Petri net is then used to provide a structured approach for qualitative and quantitative analysis of coherent and non-coherent systems. Various examples are provided to explore the behaviour of Petri net construction, modeling, and assessments. Also, limitations of Petri net simulation are introduced and solutions for modifying Petri net software classes are presented.

In the case studies, the airlock system is introduced, which has an accredited safety function to maintain containment pressure boundary. Through the understanding of Petri net construction, we were able to model the system in a structured approach and provide methods for assessment. Both qualitative and quantitative analyses are performed to obtain minimum cutsets and calculate system unavailability: $Q_t(s)$. Also, MCS is introduced in the second case study to demonstrate the qualitative analysis for a non-coherent system where the use of the inhibitor arc was necessary. This thesis has shown the successful use of Petri nets for PRA in nuclear

applications through detailed demonstration of a structured approach to Petri net construction, both qualitative and quantitative analysis, and overcoming current Petri net software limitations for simulations.

5.2 Future Work

This thesis studies the traditional Petri net and does not consider repair of failed components. Dynamic systems encompass the variables of time, failure, and repairs which more accurately describe systems in practice. Petri nets are limited by its software and the unavailability of code to model dynamic gates. While it is possible to use basic symbols to model dynamic gates, the system would immediately become too large and cumbersome for simulation and analysis. The first step to streamlining Petri nets as a viable tool for reliability assessment is to develop a software library.

It is demonstrated in this thesis that they are also a powerful tool with the ability to simulate and model systems to provide a frame by frame account of operation. Once the areas of stable software and dynamic conversions are resolved, Petri nets will be able to model and simulate system behaviour with higher accuracy.

REFERENCES

Adamyan, A., He, D. (2004, "System Failure Analysis Through Counters of Petri Net Models", Qual. Reliab. Engng. Int Vol 20, pp317-335

Ajmone, M. (1989), "Stochastic Petri nets: an elementary introduction", Springer-Verlag New York, ISBN:0-387-52494-0.

Andrews, J. D. (2000), "To Not or Not to Not!!" 18th International System Safety Conference.

Andrews, J. D. (2001), "The use of Not logic in fault tree analysis," Qual. Reliab.Eng., vol. 17, pp. 143–150.

Baccelli, F., & Canales, M. (1991), "Paralleled Simulation of Stochastic Petri Nets Using Recurrence Equations," ACM Transactions on Modeling and Computer Simulation, Vol., 3, No. 1, pp. 20-41.

Baccelli, F., Furmento, N. & Gaujal, B. (1995), "Parallel and Distributed Simulation of Free Choice Petri Nets," Proceedings of Ninth Workshop on Parallel and Distributed.

Barlow, R.E. & Proschan, F. (1965), "Mathematical Theory of Reliability (1st Ed)", Wiley, 1965.

Barlow, R.E. & Proschan, F. (1975), "Importance of system components and fault tree events," Stochastic Processes and Their Applications, vol. 3, pp. 153–173.

Bell, E. (1986), "Men of Mathematics." New York: Simon and Schuster.

Bendell, A. & Ansell, J. (1984), "The incoherency of multistate coherent systems," Reliab. Eng., vol. 8, pp. 165–178.

Birkhoff, G. & Mac Lane, S. (1996), "A Survey of Modern Algebra", 5th ed. New York: Macmillian.

Birnbaum, Z.W. (1969), "On the importance of different components in a multicomponent system, in Multivariate Analysis", P. Krishnaiah, Ed: Academic Press

Cho, S. & Jiang, J. (2008), "Analysis of surveillance test interval by Markov process for SDS1 in CANDU nuclear power plants", Reliabiilty Engineering and System Safety 93, pp 1-13.

Codetta-Raiteri, D. (2005), "The Conversion of Dynamic Fault Trees to Stochastic Petri Nets, as a case of Graph Transformation", Electronic Notes in Theoretical Computer Science 127, 45-60.

Colglazier, E., & Weatherwas, R. (1986), "Failure Estimates For The Space Shuttle", Abstracts of the society for risk analysis annual meeting, Boston, p. 80.

David, R. & Alla, H. (1992), "Petri Nets and Grafcet Tools for modeling discrete event systems", Prentice-Hall.

Dugan, J.B., Bavuso, S. K. & Boyd, M.A. (1993), "Fault Trees and Markov models for reliability analysis of fault-tolerant digital systems", Reliability Engineering and System Safety 39, 291-307.

Durga Rao, D., Gopika, V., Sanyasi Rao, V., Kushwaha, H.S., Verma, A.K., & Srividya, A. (2009), "Dynamic Fault tree analysis using Monte Carlo simulation in probabilistic safety assessment", Reliability Engineering and System Safety vol. 94.

Ericson, C. (1999), “Fault tree analysis—a history”, 17th International System Safety Conference, Orlando.

Ford, D. (1977), “A history of federal nuclear safety assessments” WASH 740. Washington: Union of Concerned Scientists.

Frankel, E. (2002), “Systems reliability and risk analysis”, 2nd ed. Boston: Kluwer Academic Publishers.

Green, A. & Bourne, A. (1972), “Reliability Technology”, London, Wiley.
IAEA-TECDOC-930 1997, “Generic component reliability data for reach reactor PSA”, Vienna, Austria.

IEC 61508, “Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES)”.

IEEE Power Engineering Society. 1985. “IEEE Guide for General Principles of Reliability Analysis of Nuclear Power Generating Station Safety Systems”, New York (NY): IEEE.

Jing, Y., Yong, L., Yanling, L. & Rui, X. (2009), “Research on Reliability modeling of complex system based on dynamic fault tree”, Technology and Innovation Conference 2009 (ITIC 2009), International, 12-14.

Kessides, I.N. (2012), “The future of the nuclear industry reconsidered: Risks, uncertainties, and continued promise”, Frontiers of Sustainability, Volume 48.

Kohda, T. (2006), “A Simple Method to Derive Minimal Cutsets for a Non-coherent Fault Tree”, International Journal of Automation and Computer 2, pp 151-156.

Kouts, H. (1998), “History of safety research programs and some lessons to be drawn from it”, 26th water reactor safety information meeting, Bethesda.

Kumanmoto, H. & Ernest J. (1978), “Top-down Algorithm for Obtaining Prime Implicant Set of Non-Coherent Fault Trees”, IEEE Transaction on Reliability, Vol. R-27, no. 4.

Kumanmoto, H. & Henley, E. J. (1996), “Probabilistic Risk Assessment and Management for Engineers and Scientists” 2nd ed., Institute of Electrical and Electronics Engineers.

Lambert, H. E. (1975), “Fault Trees for Decision Making in Systems Analysis” Ph.D., Univ. of California, Livermore.

Liu, T.S. & Chiou, S.B. (1997), “The application of Petri nets to failure analysis”, Reliability Engineering and System Safety 57, 129-142.

Lutenbach, K. (1987), “Linear Algebraic Techniques for Place/Transition Nets,” Advances in Petri Nets in Lecture Notes in Comp. Science, Springer-Verlag, pp. 142-167.

Marsan, M.A. (1989), “Stochastic Petri nets: an elementary introduction”, Springer-Verlag New York, ISBN:0-387-52494-0.

Marshavilas, P.K., Koulouriotis, D. & Gemeni, V. (2011), “Risk analysis and assessment methodologies in the work sites: On a review, classification and comparative study of the scientific literature of the period 2000-2009”, Elsevier, Journal of Loss Prevention in the Process Industries Vol 24, 477-523.

Melnyk, R.V. “Petri Nets: An Alternative to Markov Chains”, http://www.theriac.org.

Merlin, P. & Faber, D.J. (1976), "Recoverability of Communication protocols – Implications of a Theoretical Study", IEEE Transaction on Communications, vol. 24, no. 9, pp. 1036-1043.

Molloy, M.K. (1982), "Performance Analysis Using Stochastic Petri Nets", IEEE Trans. Vol C-31, pp. 913-917.

Murata, T. 1989, "Petri Nets: Properties, Analysis and Applications", Proc. of the IEEE, vol. 77, no. 4, pp. 541-580.

Murata, T. and Koh, J. Y. (1980), "Reduction and expansion of live and safe marked graphs", IEEE Trans. Circuits Syst., vol. CAS-27, no. 1, pp. 68-70, Jan. 1980.

National Reliability Evaluation Program (NREP) Procedures Guide, 1982, "NUREG/CR-2815", BNL-NUREG-51559.

NEA 1989, "Probabilistic Safety Assessment in Nuclear Power Plant Management", Report by a Group of Experts, OECD, Paris.

NEA 1991, "Living Probabilistic Safety Assessment for Nuclear Power Plant Management", Report by a Group of Experts, OECD, Paris.

Nivoliannitou, Z.S., Leopoulos, V.N. & Konstantinidou, M. (2004), "Comparison of techniques for accident scenario analysis in hazardous systems", Journal of Loss Prevention in the Process Industries 17 467-475.

NRC, 1978, "Risk Assessment Review Group Report to the U.S. Nuclear Regulatory Commission", NUREG/CR-0400.

NRC, 1979, "Statement on Risk Assessment and the Reactor Safety Study Report (WASH-1400) in Light of the Risk Assessment Review Group Report".

Ogata, K. (1987), “Discrete-Time Control Systems”, Prentice-Hall, N.J.

Peterson, J. (1981), “Petri Net Theory and the Modeling of Systems”, Prentice-Hall, Inc, N.J., USA.

Petri, C., & Kommunikation mit Automaten. (1962), “Communication with Automata”, University of Bonn, West Germany.

PRA Procedures Guide, 1983, NUREG/CR-2300.

Rasmuson, D. M., Shepherd, J. C., Marshall, N. M., & Fitch, L.R. (1982), “Use of COMCAN III in system design and reliability analysis”, Report EGG-2187, EG&G Idaho.

Rausand, M. & Hoyland, A. (2003), System Reliability Theory: Models, Statistical Methods, and Applications. 2n^{d} ed. Wiley.

Reactor Safety Study, 1975, “An Assessment of Accident Risks in U.S. Commercial Nuclear Power Plants”, WASH-1400, NUREG-75/014.

Reisig, W. (1985), “Petri Nets: An Introduction”, Springer-Verlag.

Reisig, W. (1992), “Combining Petri Nets and Other Formal Methods”, Application and Theory of Petri Nets, Lecture Notes in Computer Science, Springer-Verlag, Vol. 616, pp.24-44.

Relex, 2007, “Reliability Studio Reference Manual”, Relex Software Corporation, Greensburg, PA USA.

Rhodes, R. (1986), “The making of the atomic bomb”, New York, Simon and Schuster.

Rohr, C., Marwan, W. & Heiner, M. (2010), "Snoopy -–a unifying Petri net framework to investigate biomolecular networks", Bioinformatics 26, 7: 974-975.
Sandia National Laboratories, 1985, "1985 Simulation", Albuquerque, New Mexico, France, pp. 3-10.

Sifakis, J. (1980),"Performance Evaluation of Systems Using Nets," Net Theory and Applications, Proc. Of Advanced Course on General Net Theory of Processes and Systems, Lecture Notes in Computer Science, Springer-Verlag, vol. 84, pp. 307-319.

Silva, M. (1985), "Petri Nets in Automation and Computer Engineering", Madrid, Spain, Editorial AC, (in Spanish) 1985; English translation to be published by Kluwer Academic Press, Hingham, MA.

The international Marine Contractors Association 2002. Guidance on Failure Modes & Effects Analyses (FMEAs).

US NRC, 1978, "Statement on risk assessment and the reactor safety study report".

US NRC. A short history of nuclear regulation, 1946–1999 (NUREG/BR-0175, Rev 1)

US NRC. NUREG/CR-1659, 1981, "Reactor safety study methodology applications program", US Nuclear Regulatory Commission, (Vol. 1), (Vol. 2), (Vol. 3), (Vol. 4).

USNRC, 1980, "Special Group Inquiry; Three Mile Island – A Report to the Commissioners and to the Public".
Worrell, R.P. (1985), "SETS Reference Manual", NUREG/CR-4213, SAND832675.

Zhang, X., Miao, Q., Fan, X. & Wang, D. (2009), "Dynamic Fault Tree Analysis based on Petri Nets", Reliability, Maintainability and Safety.

Appendix A: Event Tree Analysis

The ETA method (Event Tree Analysis) is a technique that uses decision trees and logically develops visual models of the possible outcomes of an initiating event. It is a graphical representation of the logic model that identifies and quantifies the possible outcomes following the initiating event [IEEE Std 352, 1987]. In this method, an initiating event such as the malfunctioning of a system, process, or construction is considered as the starting point. The predictable accidental results, which are sequentially propagated from the initiating event, are presented in order graphically.

ETA is a system model representing system safety based on the safeties of sub-events. It is called an event tree because the graphical presentation of sequenced events grows like a tree as the number of events increase. An event tree consists of an initiating event, probable subsequent events, and final results caused by the sequence of events. Probable subsequent events are independent to each other and the specific final result depends only on the initiating event and the subsequent events following. Therefore, the occurrence probability of a specific path can be obtained by multiplying the probabilities of all subsequent events existing in a path. In an event tree, all events in a system are described graphically and it is very effective to describe the order of events with respect to time because the tree is related to the sequence of occurrences. In the design stage, ETA is used to verify the criterion for improving system performance; to obtain fundamental information of test operations and management and to identify useful methods to protect a system from failure. The ETA technique is applicable not only to design, construction, and operation stages, but also to the change of operation and the analysis of accident causes. The modeling limitation of ETA is the inability to accurately model repairs or replacements. A

typical event tree (ET) is represented below where the initiating event, seen on the left as explosion, has just occurred and the tree grows with each new event.

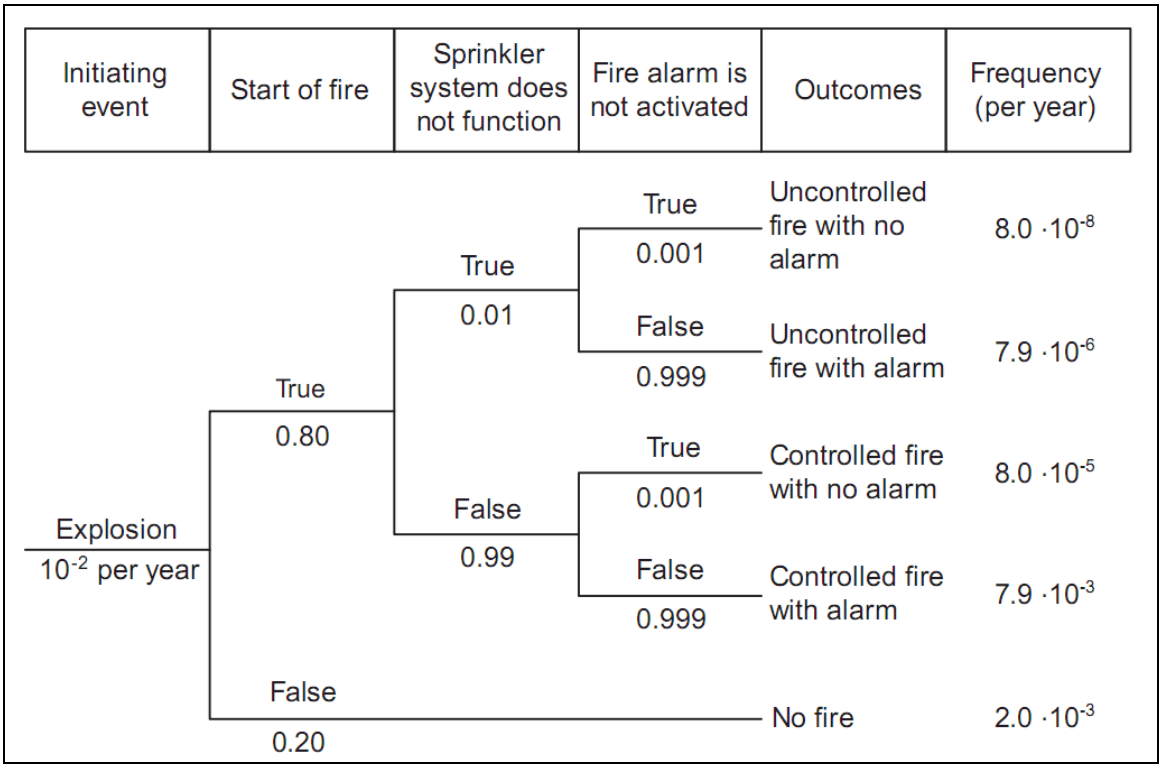


Figure 36 ETA Model [Rausand & Hoyland, 2003]

The main steps to building an event tree are as follows:

1. Identify a relevant accident event that may give rise to unwanted consequences
2. Identify the barriers that are designed to deal with the accident event
3. Construct the event tree
4. Describe the resulting accident sequences
5. Determine the frequency and the probabilities of the branches in the event tree
6. Calculate the probabilities/frequencies for the identified consequences
7. Compile and present the results from the analysis

Appendix B: Petri Net Interaction Reduction Rules

Often when dealing with a large system, it is necessary to reduce the system mode to a simpler one while preserving the system properties. In this section only the simplest transformations are presented [Silva, 1985] [Murata & Koh, 1980].

Let (N, M_0) and (N', M_0') be the Petri nets before and after one of the following transformations.

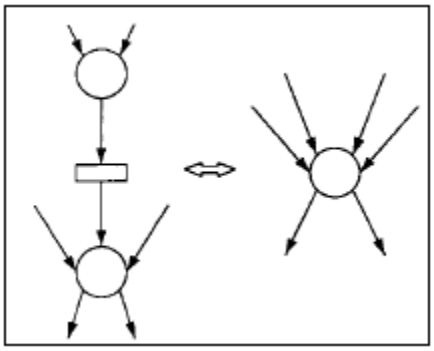


Figure 37 Fusion of Series Places (FSP) [Murata & Koh, 1980]

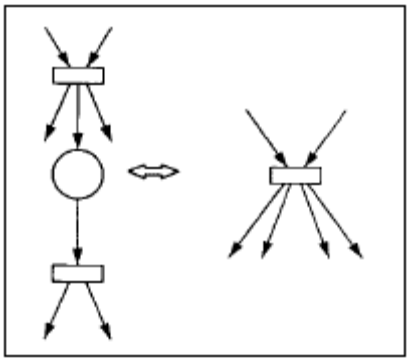


Figure 38 Fusion of Series Transitions (FST) [Murata & Koh, 1980]

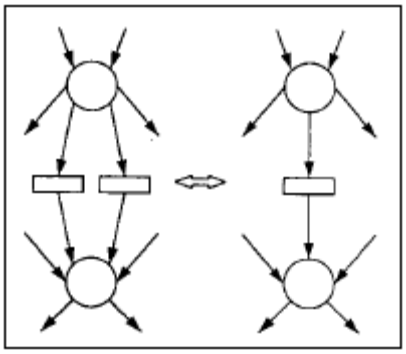


Figure 39 Fusion of Parallel Transitions (FPT) [Murata & Koh, 1980]

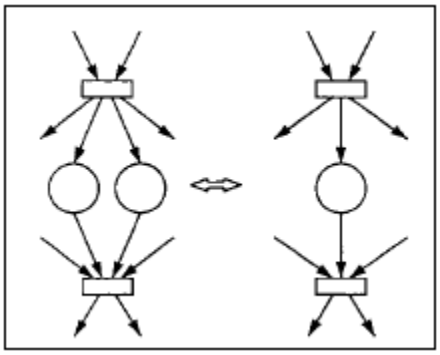


Figure 40 Fusion of Parallel Places (FPP) [Murata & Koh, 1980]

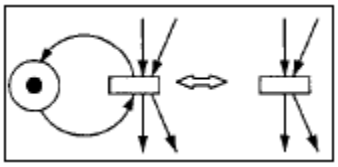


Figure 41 Elimination of Self-loop Places (ESP) [Murata & Koh, 1980]

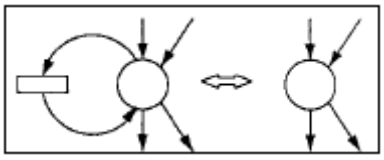


Figure 42 Elimination of Self-loop Transitions (EST) [Murata & Koh, 1980]

Appendix C: Non-coherent System Properties

A fault tree is non-coherent if its structure function Φ ($\underline{x}$) does not comply with the definition of coherence given by the properties of relevance and monotonicity [Barlow & Proschan, 1965], [Bendell & Ansell, 1984]. A non coherent Fault Tree will consist of basic events which are in a non failed state.

Φ ($\underline{x}$) - structure function: defines the system state in terms of the states of the system components

1) Every component *i* is relevant: $\Phi(1_i, \underline{x}) \neq \Phi(0_i, \underline{x})$ for some $\underline{x}$.

2) The structure function of component *i* is monotonically increasing: $\Phi(1_i, \underline{x}) \geq \Phi(0_i, \underline{x})$, for all *i* and $\underline{x}$.

$$\Phi(1_i, \underline{x}) \equiv \Phi(x_1,...,x_{i-1}, 1, x_{i+1}, ..., x_n);$$

$$\Phi(0_i, \underline{x}) \equiv \Phi(x_1,...,x_{i-1}, 0, x_{i+1}, ..., x_n).$$

• Condition #1 ensures that each component contributes to the system state.

• Condition #2, an increasing (non-decreasing) structure function, ensures that the system state deteriorates (or at least does not improve) with an increasing number of component failures. Component failures cannot improve the system state.

Appendix D: Software Development Source Code

```
/************************
Place Class
************************/

import java.awt.Point;

public class Place {
  protected int tokens ;
  protected int bound ;
  protected Boolean bound_state;
  protected String name ;
    protected Point coord;

  public Place() {
    this(0) ;
  }

  public Place(int t) {
    this(t,-1) ;
  }

  public Place(int t, int b, b   odelinc) {
    tokens = t ;
    bound = b ;
    bound_state = c;
    coord = new Point();
  }

  public void setName(String name) {
    this.name = name;
  }

  public String getName() {
    return name;
  }

    public void setCoord(int x, int y) {
          coord.setLocation(x,y);
    }
}
```

```
/*************************
Arc Class
*************************/
class Arc {
  public Place place ;
  public int weight ;
  public int bound ;

  public Arc(Place p) {
    this(p,1) ;
  }

  public Arc(Place p, int w) {
    place = p ;
    weight = w ;
    bound = p.bound ;
  }
}
/*************************
Transition Class
*************************/

import java.util.HashSet ;
import java.awt.Point;

public abstract class Transition {
  protected HashSet in, out ;
  protected b   odelinimmediate ;
  protected String name ;
    protected Point coord;

  public Transition() {
    immediate = true ;
    in = new HashSet() ;
    out = new HashSet() ;
    coord = new Point();
  }

    public abstract void onFire() ;

  public void addIn(Place p) {
    addIn(p,1) ;
  }

  public void addIn(Place p, int weight) {
    synchronized(in) {
```

```
      in.add(new Arc(p,weight)) ;
    }
  }

  public void addOut(Place p) {
    addOut(p,1) ;
  }

  public void addOut(Place p, int weight) {
    synchronized(out) {
      out.add(new Arc(p,weight)) ;
    }
  }

  public void setImmediate(b   odelinimm) {
    immediate = imm ;
  }

  public void setName(String name) {
    this.name = name ;
  }

  public String getName() {
    return name;
  }
}
```