UNIVERSITY OF Hull

Compositional Dependability Analysis of Dynamic Systems with Uncertainty

being a Thesis submitted for the Degree of
Doctor of Philosophy
in the University of Hull

by

Sohag Kabir BSc., MSc.

February 2016

Abstract

Over the past two decades, research has focused on simplifying dependability analysis by looking at how we can synthesise dependability information from system models automatically. This has led to the field of model-based safety assessment (MBSA), which has attracted a significant amount of interest from industry, academia, and government agencies. Different model-based safety analysis methods, such as Hierarchically Performed Hazard Origin & Propagation Studies (HiP-HOPS), are increasingly applied by industry for dependability analysis of safety-critical systems. Such systems may feature multiple modes of operation where the behaviour of the systems and the interactions between system components can change according to what modes of operation the systems are in.

MBSA techniques usually combine different classical safety analysis approaches to allow the analysts to perform safety analyses automatically or semi-automatically. For example, HiP-HOPS is a state-of-the-art MBSA approach which enhances an architectural model of a system with logical failure annotations to allow safety studies such as Fault Tree Analysis (FTA) and Failure Modes and Effects Analysis (FMEA). In this way it shows how the failure of a single component or combinations of failures of different components can lead to system failure. As systems are getting more complex and their behaviour becomes more dynamic, capturing this dynamic behaviour and the many possible interactions between the components is necessary to develop an accurate failure model.

One of the ways of modelling this dynamic behaviour is with a state-transition diagram. Introducing a dynamic model compatible with the existing architectural information of systems can provide significant benefits in terms of accurate representation and expressiveness when analysing the dynamic behaviour of modern large-scale and complex safety-critical systems. Thus the first key contribution of this thesis is a methodology to enable MBSA techniques to model dynamic behaviour of systems. This thesis demonstrates the use of this methodology using the HiP-HOPS tool as an example, and thus extends HiP-HOPS with state-transition annotations. This extension allows HiP-HOPS to model more complex dynamic scenarios and perform compositional dynamic dependability analysis of complex systems by generating Pandora temporal fault trees (TFTs). As TFTs capture state, the techniques used for solving classical FTs are not suitable to solve them. They require a state space solution for quantification of probability. This thesis therefore proposes two methodologies based on Petri Nets and Bayesian Networks to provide state space solutions to Pandora TFTs.

Uncertainty is another important (yet incomplete) area of MBSA: typical MBSA approaches are not capable of performing quantitative analysis under uncertainty. Therefore, in addition to the above contributions, this thesis proposes a fuzzy set theory based methodology to quantify Pandora temporal fault trees with uncertainty in failure data of components.

The proposed methodologies are applied to a case study to demonstrate how they can be used in practice. Finally, the overall contributions of the thesis are evaluated by discussing the results produced and from these conclusions about the potential benefits of the new techniques are drawn.

To my parents and beloved wife ⋯

উৎসর্গ

আমার মা-বাবা এবং প্রিয়তমা স্ত্রীকে

Acknowledgements

The journey of the PhD is complete, the mission is at its end. But I admit that without the blessings and protection of almighty Allah and support from many people the mission could not have been achieved.

First and foremost, my heartfelt gratitude, profound indebtedness and deep respect go to my supervisor Dr Martin Walker for his constant supervision, affectionate guidance and great encouragement and motivation. His keen interest on the topic and valuable advice throughout my study was of great help in completing the thesis. Without him the achievement could never have come this far.

I am also very grateful to Professor Yiannis Papadopoulos for his support, enthusiastic encouragement and insightful advice that helped me to shape this thesis. I would like to thank Dr Darryl Davis, chair of my panel meetings, for his constructive criticisms and insightful advice which contributed to the improvement of the overall project.

I would also like to thank all the staff members in the Department of Computer Science especially Dr Neil Gordon, Dr Septavera Sharvia, Dr David Parker, Dr Leonardo Bottaci, Simon Grey, Amanda Millson, Helen El-Sharkawy, Lynn Morrell, Jo Clappison, Sally Byford, Adam Hird, Andrew Hancock, Mark Bell, and Mike Bielby, for their academic, administrative and technical support. I also like to express my sincere gratitude to Dr Marco Cattaneo for his expert opinion on the decision making process under uncertainty and Dr Marco Paolieri for his help with the ORIS tool.

I would like to acknowledge and extend my heartfelt gratitude to my fellow colleagues, who made the PhD experience more bearable through their support and friendship in good and bad times. I say thank you — Luis Azevedo, Dr Ernest Edifor, Dr Zhibao Mian, Dr Lamis Farah Al-Qora'n, Ioannis Sorokos, Youcef Gheraibia, John Dixon, Sahar Arshi, Seyed Sadegh Zadeh, John Stamford, Luis Torrao, Dr Lisa Moore, Tareq Aljaber, Mohammad Al Khaldy, Qian Wang, Dr Jose Aizpurua, Dr Mustafizur Rahman, Dr Nidhal Mahmud, and Dr Nabil Abu Hashish.

I would like to say a big thank you to all my friends and relations in Hull and abroad who have supported me throughout my PhD in various ways: Ashraful Alam, Risalat Mahmud, Badrul Haider, Enamul Haque, Mokhlesur Rahman, Syed Mohammed Hassan, Khairul Bashar, Rizwan Ahmed, Tanzima Azad, Muksud Shanto, Rashedul Hassan, Saif Hasib, Atiqul Islam, Fahad Ibrahim, Salman Buksh, Faisal Miaze, Masrurul Mowla, Zuhair Yusuf, Ehtesham, and Sayed Al-Fazari. I wish to thank all other friends, classmates, and colleagues who perhaps deserve a credit for their help but aren't listed here.

The entire PhD research was fully supported by a University of Hull PhD Scholarship. I am very grateful to the University of Hull for this financial support without which the dissertation would have been impossible.

Finally, I would like to give my deepest thanks to my parents, my wife, my brothers, my sister, my in-laws and other relatives for their continued support and encouragement throughout my study and for teaching me many useful lessons for life.

Author's Declaration

I declare that the material contained in this thesis represents original work undertaken solely by the author. The various aspects of the work covered in this material have been presented in a number of international conferences and scientific publications.

The work on fuzzy set theory based quantification method for temporal fault trees in Chapter 4 was presented in Kabir *et al.* (2014a); another article on this topic has been extracted from this chapter and submitted for publication. The work on the conversion of Pandora temporal fault trees into Bayesian Networks (Chapter 4) was presented in Kabir *et al.* (2014b). The Petri Net based methodology for probabilistic evaluation of Pandora temporal fault trees (Chapter 4) was presented in Kabir *et al.* (2015). Parts of the literature review of the thesis (Chapter 2) have been published as a book chapter in Sharvia *et al.* (2015).

List of Abbreviations

AADL	Architecture Analysis and Design Language
BFS	Behavioural Fault Simulation
CFT	Component Fault Tree
CPT	Conditional Probability Table
CTBN	Continuous Time Bayesian Network
CTMC	Continuous Time Markov Chain
DBN	Dynamic Bayesian Network
DCCA	Deductive Cause Consequence Analysis
DFT	Dynamic Fault Tree
DSPN	Deterministic Stochastic Petri Net
ELRAFT	Efficient Logic Reduction Analysis of Fault Trees
FFPS	Fuzzy Failure Possibility Score
FFTA	Fuzzy Fault Tree Analysis
FIM	Fuzzy Importance Measure
FLSA	Failure Logic Synthesis and Analysis
FMEA	Failure Modes and Effects Analysis
FMECA	Failure Modes Effects and Criticality Analysis
FPTC	Failure Propagation and Transformation Calculus
FPTN	Failure Propagation and Transformation Notation
FSAP	Formal Safety Analysis Platform
FTA	Fault Tree Analysis
GSPN	Generalized Stochastic Petri Net
HBN	Hybrid Bayesian Network
HiP-HOPS	Hierarchically Performed Hazard Origin & Propagation Studies
MBSA	Model Based Safety Assessment
MCS	Minimal Cut Set
MCSQ	Minimal Cut Sequence
MOCUS	Method of Obtaining Cut Sets
MTBF	Mean Time Between Failure
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
RTN	Real Time Network
SAML	Safety Analysis Modelling Language
SEFT	State Event Fault Tree
TFT	Temporal Fault Tree
ViSSaAn	Visual Support for Safety Analysis

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Field of Research and Problem Context

Safety-critical systems have an intense effect on almost every aspect of our life (e.g. manufacturing, communications, medicine, education, defence, financial, amusement, research, etc.). Systems that fall into this category range from propulsion systems on spacecraft to airbags in cars; however, they all share a common property — their failure has the potential to cause great harm to people. Therefore, dependability of critical systems is a prime concern in modern society due to our increasing dependence on those systems. Dependability is the capability of avoiding failures that are more frequent and more severe than is acceptable, and thus dependability assessment should be carried out early in the design phase to avoid unacceptable costs in terms of loss of life, environmental damage, and loss of resources by identifying and rectifying potential hazards as soon as possible (Bernardi and Merseguer, 2007). The dependability of a system includes, but is not limited to the following characteristics: safety, reliability, and maintainability.

One of the key goals in designing safety-critical systems is to identify potential risks posed by such systems and then minimising the likelihood of these risks. This necessity has led to the field of reliability engineering, which concentrates on making systems as reliable and as safe as possible for the stakeholders. To make systems more reliable the analysts have to understand the behaviour of the systems, i.e., how systems work and how they may fail. Systems analysis is a process that allows reliability engineers to understand how systems work and how they can fail by investigating the system behaviour and potential causes of system failure, thereby allowing them to determine necessary actions to prevent system failure (Vesely *et al.*, 2002). There are generally two forms of analysis. The first is qualitative analysis, which helps the analysts to determine the necessary and sufficient causes or combinations of causes that can lead the system to failure. The second is quantitative analysis, which helps to estimate the probability that the system will fail after a specified amount of time given the probability of the basic component failure. It also helps to determine which components or parts of the system are more critical so analysts can put more emphasis on the critical components or parts

by taking necessary steps, e.g., including redundant components in the system model.

There are many widely used classical safety assessment methods available to assist safety analysts in performing dependability analysis of systems. One such widely used method is Failure Modes Effects and Criticality Analysis (FMECA). It was originally specified in US Military Procedure MIL-P-1629 and then updated in MIL-STD-1629A (US, 1980). It is an inductive analysis method that considers all possible combinations of effects of a single component failure mode(s). This method also provides ways to perform probabilistic analysis to determine criticality of failure modes. Another well-established classical safety analysis method is Fault Tree Analysis (FTA) (Vesely *et al.*, 2002). It is possibly the most widely used method for evaluating safety and reliability of static systems qualitatively as well as quantitatively. Static systems are those which only experience a single mode of operation throughout the duration of their lifetimes, and thus exhibit constant nominal and failure behaviours.

Fault trees utilise graphical representations based on Boolean logic to show logical connections between different faults and their causes (Vesely *et al.*, 2002). FTA is a deductive analysis method, which means analysis starts with a system failure known as the 'top event' and works backwards to determine its root causes. From a fault tree, it is possible to understand how combinations of failures of different components or certain environmental circumstances can lead to system failure. After construction of a fault tree, qualitative analysis is performed using Boolean logic by reducing it to minimal cut sets (MCSs), which are a disjoint sum of products consisting of the smallest combinations of failure events that are necessary and sufficient to cause the top event. It is worth noting that inductive and deductive methods are often complementary and effective in different circumstances. For example, inductive methods may be used to identify system hazards or to verify the completeness of previously identified hazards; however, usually they are not capable of considering complex failure scenarios. On the other hand, deductive methods cannot identify or verify system hazards, but they can systematically identify failure scenarios containing a large number of independent failure modes.

All these classical techniques, regardless of their logical orientation (inductive, deductive), are often primarily manual processes, i.e., performed manually either by a single person or a group of persons to produce some comprehensive documents to satisfy the safety requirements and to determine strategies to alleviate the effects of failures (Leveson, 1995). Although these techniques can produce a significant amount of valuable knowledge about the system safety and reliability, due to their manual nature they have some limitations. Firstly, in the manual process these analyses are performed based on the informal knowledge of the failure behaviour of the systems, whereas the architecture of the system is modelled more formally; therefore these processes can result in inconsistencies and discrepancies. Secondly, as the system grows in size, the manual nature of the analysis process increases the risk of introducing error or producing incomplete results. Moreover, the manual analyses are time consuming and expensive, therefore the

analyses are rarely carried out more than once even though the iterative process could produce more valuable information. Finally, the informal nature of these analyses does not allow a high degree of reusability of information, i.e., if a new analysis is required on the previously designed system or the system design is changed a bit, then the analysis is typically required to be started from the beginning meaning that it is difficult to reuse materials from a previous analysis.

To overcome the above mentioned limitations, a new field of model-based safety assessment (MBSA) has emerged (Walker *et al.*, 2008). MBSA has attracted significant interest in the industry and academia over the last twenty years. In MBSA, the analysts perform their analyses on the design model of the system, which is created as part of a model-based design process. As the analyses are performed on a more formal model rather than a separate safety analyses model, it opens the avenue to automate part of the safety analysis process, e.g., automatically generating fault trees. In MBSA, as the analyses are performed on formal models, the analyses can be performed iteratively, which helps to generate more results and new results can be generated if the system design changes. This process is less time consuming and less expensive compared to manual approaches and due to its more structured nature, the risks of introducing errors in the analysis or producing incomplete results are reduced. Moreover, the MBSA techniques provide a higher degree of reusability by allowing parts of an existing system model, or libraries of previously analysed components, to be reused.

MBSA techniques can be classified into two broad categories (Walker *et al.*, 2008) based on their general underlying formalism and the types of analysis performed. The first paradigm is called Failure Logic Synthesis and Analysis (FLSA) (Sharvia and Papadopoulos, 2015) which focuses on the automatic construction of predictive system analyses. These approaches are typically compositional, meaning that system-level failure analyses can be generated from component-level failure logic and the topology of the system. In these approaches, the system-level analysis and assessment are broken down into more manageable tasks, applied to individual components, and a system level failure model is then produced by composing the failure models of individual components. The composition of individual component failure models is usually done by connecting output deviations of a component to input deviations of other components. This compositionality lends itself well to automation and reuse of component failure models across applications. In order to simplify and automate the synthesis process, several tools and techniques such as Hierarchically Performed Hazard Origin & Propagation Studies (HiP-HOPS) (Papadopoulos, 2012), Component Fault Trees (CFT), and the Failure Propagation and Transformation Notation (FPTN) have emerged as part of FLSA. These techniques usually combine different classical safety analysis approaches such as FTAs or FMEAs to allow the analysts to perform safety analyses automatically or semi-automatically. Using the classical approaches, the FLSA tools or techniques can determine how a single component failure or combinations of failures of different components or certain environ-

mental circumstances can lead to system failure. This information may be sufficient for the analysis of some systems (static systems) but may not be enough for many systems with complex architectures.

The second paradigm is called Behavioural Fault Simulation (Walker *et al.*, 2008) which focuses on behavioural simulation to automatically analyse potential failures in a system. This development has led to a group of formal verification based approaches. The techniques that fall into this category include the Formal Safety Analysis Platform/New Symbolic Model Verifier (FSAP-NuSMV) (Bozzano and Villafiorita, 2007), AltaRica (Arnold *et al.*, 2000; Point and Rauzy, 1999), and Deductive Cause Consequence Analysis (DCCA) (Güdemann *et al.*, 2007; Ortmeier *et al.*, 2005). These approaches generally use state-event based formalisms and work by injecting possible faults into simulations based on executable, formal specifications of a system and studying the effects of those faults on the system behaviour. The results are then used by model checking tools to verify whether system dependability requirements are being satisfied or whether violations of the requirements exist in normal or faulty conditions.

One important characteristic of systems with complex architecture is their dynamic behaviour, i.e., the behaviour of the system (both nominal and potential failure behaviour) can change according to what state or mode of operation the system is in. As the behaviour of the system changes, functions and their failure modes vary, as do the flows between components of the architecture and the potential deviations of those flows. Due to this complex behaviour and the many possible interactions between the components, assessing the effects of combinations of failure events is not enough by itself to capture the system failure behaviour; in addition, understanding the order in which they fail is also required for a more accurate failure model.

However, this dynamic system behaviour creates difficulties, e.g., in capturing sequence-dependent behaviour for compositional FLSA techniques when using classical safety analysis techniques like FTA or FMECA as their primary means of analysis (Bruns and Anderson, 1993; Dugan *et al.*, 1990; Papadopoulos, 2000; Walker and Papadopoulos, 2009). This is because of the absence of the concept of state in the classical safety analysis techniques.

As FTA is the most widely used classical safety and reliability analysis technique and the limitations of FTA are not new, there are already a number of different modifications proposed to overcome those limitations. For example, the Dynamic Fault Tree (DFT) (Dugan *et al.*, 1992) approach is an extension of classical FTs. It introduces dynamic gates such as Priority-AND (PAND), Functional Dependency (FDEP) and Sequential Enforcing (SEQ) to capture sequence dependent behaviour. The primary focus of DFTs is the quantitative analysis of the failure behaviour of the dynamic systems. DFTs are usually quantified by converting them to equivalent Markov chains (MCs). Sequence-dependent behaviour can then be readily modelled using Markov models, but the major shortcoming of a Markov model is that for large and complex systems the Markov model

suffers increasingly from combinatorial explosion and is therefore difficult to construct and analyse. Another problem of using MCs is that they perform the analysis based on the exponential distribution of the failure rates of components, therefore components having failure rates with any other distribution cannot be analysed with this method.

Pandora (Walker, 2009) is another extension of fault trees which introduces three temporal gates and a set of temporal laws to allow qualitative analysis of dynamic systems. It was created to allow the MBSA techniques to synthesise and analyse temporal fault trees from the behavioural architecture of the dynamic systems instead of synthesising and analysing classical fault trees. The primary intention behind creating this extension of FTs was to perform qualitative analysis of dynamic behaviours of systems. Qualitative analysis can produce important dependability related information about a system; however, quantitative analysis is also important to understand the probabilistic information about the dependability of the system. Recently, some efforts have been made to quantify Pandora temporal fault trees based on exponential failure rates of components.

However, those techniques do not consider uncertainty in failure data, uncertain component behaviour, and components with non-exponentially distributed failure data. As the outcome of quantitative analysis is entirely dependent on the precision of the numerical values used in the analysis, if uncertainties are left unresolved then there is a chance of producing misleading results. A significant amount of researches (e.g., (Tanaka *et al.*, 1983; Suresh *et al.*, 1996; Ferdous *et al.*, 2009; Tyagi *et al.*, 2010)) have been performed to address uncertainty in failure data in FTA. Researchers usually use fuzzy set theory based methodology to capture uncertainty during the quantitative analysis of fault trees, and they refer this method as fuzzy fault tree analysis (FFTA). Although a significant amount of research has investigated how to use fuzzy set theory in classical FTA to enable it to perform quantitative analysis with limited quantitative data, limited researches such as Li *et al.* (2012, 2015); Verma *et al.* (2006); Yang (2011) have been undertaken to allow the same in dynamic fault tree analysis. The use of Bayesian Networks to perform quantitative analysis of uncertainty in FTA and DFT is also found in the literature (e.g., Bobbio *et al.* (2001); Montani *et al.* (2008); Boudali and Dugan (2005)).

Another approach is to use an alternative representation of failure behaviour instead of fault trees, one capable of better taking the concept of state into account. Consequently, state transition formalism based methodologies, e.g., state machines, are increasingly being used to capture dynamic behaviour of systems due to their simplicity. State machines have been used in several techniques, including but not limited to: Architecture Analysis and Design Language (AADL) (Feiler and Rugina, 2007; Feiler *et al.*, 2006b), FSAP/NuSMV-SA (Bozzano and Villafiorita, 2003), and AltaRica (Fan *et al.*, 2011; Rauzy, 2002). Some preliminary ideas on a hierarchical system modelling based on state machines in HiP-HOPS are provided by Papadopoulos (2000) (see Figure 1) but no guidelines were provided to show how the component annotations can be done and no rules were formally defined to show how communications between different levels in the

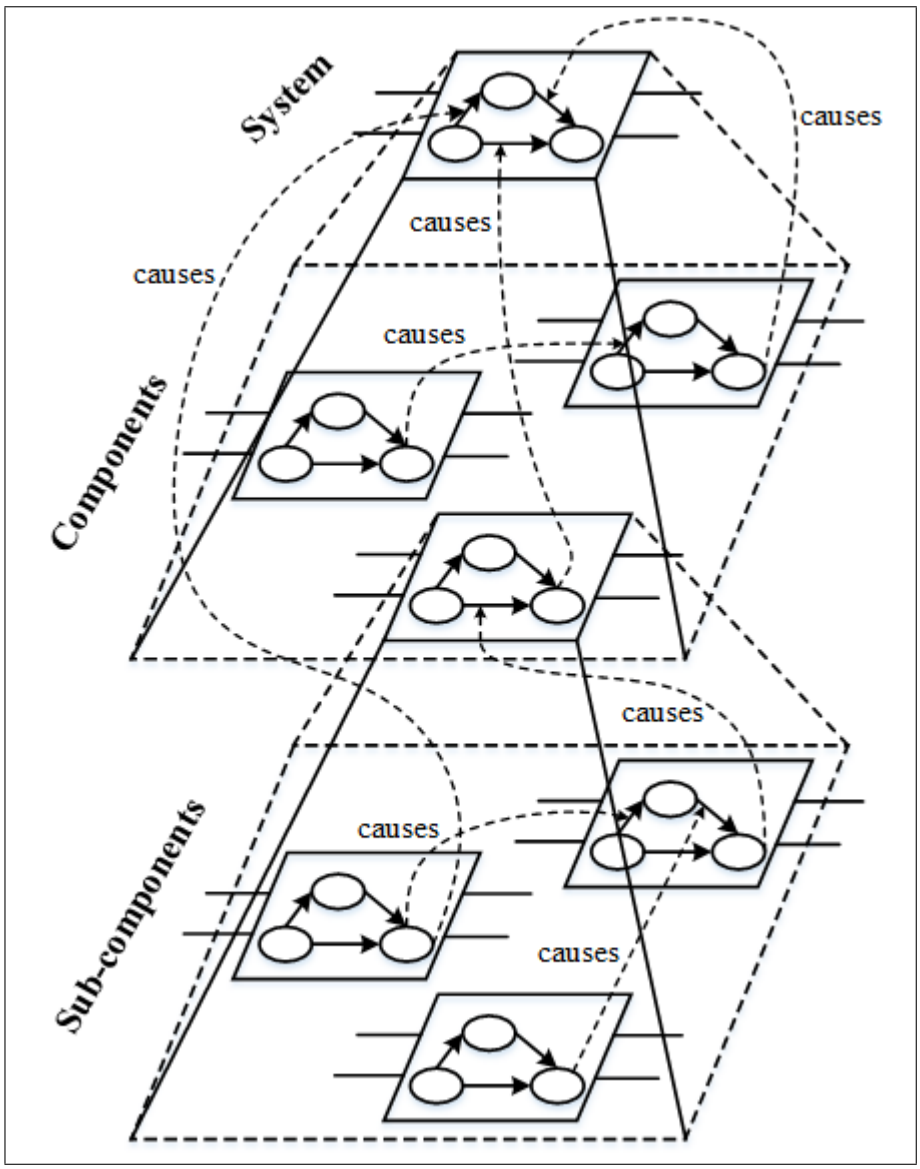


Figure 1: Representation of system behaviour as hierarchy of state machines (Papadopoulos, 2000)

hierarchy will take place. At present, the hierarchical model proposed by Papadopoulos (2000) allows partial or local evaluation of the system, but the reliability evaluation of the whole system is not enabled.

1.2 Motivation and Scope

1.2.1 Challenges in MBSA

It is clear that systems are getting more complex with the passing of time and their behaviour is becoming more dynamic. This complexity presents many challenges for the analysts, making it difficult to perform meaningful analyses to evaluate the safety and reliability of such systems. Many of the techniques that are widely used for evaluating system safety and reliability, such as FTA, are primarily manual, static approaches. MBSA is

an emerging field which provides tools and techniques to help automate these approaches. But there exist many developments and challenges in this area. One such problem is that of developing fault propagation models that link causes to effects. Another challenge is optimisation of system architectures to meet requirements or to allocate requirements.

One of the state-of-the-art MBSA techniques that can resolve this is HiP-HOPS, which combines fault trees (FTs) and FMEAs to allow a semi-automated safety analysis of complex systems. In this technique, fault trees and FMEA are automatically generated from topological system models that have been annotated with component failure behaviour. Component failure behaviour is described as a set of Boolean expressions showing how the deviations in the component outputs can be caused either by internal failure of that component or corresponding deviations in the component's input. As it uses both fault trees and FMEA, analysts can benefit from the strength of both inductive and deductive analysis approaches. This is the only tool that is considered to be capable of automatically generating FMEA from fault trees (Grunske and Han, 2008). The capability of the HiP-HOPS tool can be summarised as: *"If we know the 'structure' of a system (model) and the 'local failure behaviour of its components' (IF-FMEAs) then we can mechanically derive the 'failure behaviour of the system' (fault trees)"* (Papadopoulos, 2000, p.86). The contributions of the HiP-HOPS tool to the field of model based safety analysis are many and the important ones are:

1. Fast algorithms for automatic generation of Fault Trees and Failure Modes and Effects Analyses (FMEAs) as part of system dependability analysis.

2. Multi-objective optimisation of system architecture using genetic algorithms.

3. Novel algorithms for semi-automatic allocation of safety requirements to the system components in the form of Safety Integrity Levels.

As already mentioned, Failure Logic based compositional MBSA techniques are not typically capable of capturing sequence-dependent dynamic behaviour of systems because of their reliance on classical analysis approaches, e.g. FTA or FMECA. For this reason, the application of HiP-HOPS is limited to static models of systems. However, as mentioned earlier, State Machines or state–transition formalisms are increasingly being used in model-checking based MBSA approaches as a means for capturing dynamic behaviour of systems. The question therefore is: how can state machines be integrated into FLSA techniques (such as HiP-HOPS) to gain some of the benefits enjoyed by the model-checking approaches, i.e., in terms of modelling dynamic behaviour? This is the first motivation of this thesis.

The solution to this question requires the formulation of methodologies to annotate system components with dependability related information using state-event based techniques in addition to the Boolean logic based annotations, thus allowing dynamic analysis based on these annotations. Such a solution should be generalisable to any compositional MBSA tool or technique and should therefore contribute to the overall MBSA paradigm.

One issue with state machines is that it is not readily possible to analyse them directly. They must first be translated to other representations, e.g. temporal fault trees or Bayesian Networks, to facilitate meaningful analysis. Research has been performed to translate state machines to classical fault trees, but such a transformation does not preserve the sequence dependent behaviour of the system. However, recently, Mahmud *et al.* (2012) have introduced ways of translating state machines into Pandora temporal fault trees. Therefore, if state-based behavioural annotations could be incorporated in the MBSA techniques, then Pandora temporal fault trees can be generated from the state machines to show the failure behaviour of the system.

1.2.2 Challenges in Quantitative Analysis of Dynamic Systems using Pandora

Pandora can be used to determine minimal cut sequences (MCSQs) of TFTs, which are the smallest sequences of events that are necessary and sufficient to cause the top events, analogous to minimal cut sets (MCSs) of classical fault trees. One of the advantages of Pandora over other dynamic counterparts (e.g. DFT) is that it can perform qualitative analysis, allowing it to provide useful insight into system failures with limited or absent quantitative failure data, e.g., in the case of new system components. Moreover, the technique is integrated well with model-based design and analysis. It has been shown by Walker and Papadopoulos (2009) that Pandora logical expressions can be used to describe the local failure behaviour of components and then enable compositional synthesis of TFTs from systems models using popular modelling languages, e.g. Matlab Simulink, EAST-ADL, or AADL, that have been annotated with Pandora expressions.

Pandora is different from Dynamic Fault Trees in that it was developed with an aim to facilitate only the qualitative analysis of the systems, whereas DFTs are primarily used for quantitative analysis. Although the qualitative analysis can produce important information about the dependability of the systems, it is advantageous (and sometime necessary) to have quantitative information about the dependability of the system. As the dynamic extensions of the fault trees model the state-based behaviour of the system, the combinatorial quantification techniques used to quantify classical fault trees cannot be applied to quantify the dynamic versions of the fault trees. Various techniques have been already developed to quantify DFTs, such as Markov chain-based methods (Boudali *et al.*, 2007, 2010; Dugan *et al.*, 1993), analytical approaches (Merle *et al.*, 2011, 2014, 2010), Monte Carlo simulation based methods (Ejlali and Miremadi, 2004; Rao *et al.*, 2009; Zhang and Chan, 2012), Bayesian Network based approaches (Boudali and Dugan, 2005, 2006; Montani *et al.*, 2008; Neil *et al.*, 2008), Petri Net based approaches (Codetta-Raiteri, 2005; Zhang *et al.*, 2009), compositional approaches (Chiacchio *et al.*, 2013, 2011), and a modularisation approach (Gulati and Dugan, 1997).

Although a number of methods are available to quantify DFTs, very limited research has been performed to quantify Pandora temporal fault trees. For instance, recently an

analytical method (Edifor *et al.*, 2012, 2013) and a Monte Carlo simulation-based method (Edifor *et al.*, 2014) to quantify Pandora temporal fault trees have been developed. Out of these two methods, the analytical method is only applicable to systems with exponentially distributed failure data and the simulation-based method is computationally expensive and time consuming.

The increasing importance of model-based design and analysis, the potential use of Pandora in this context, and different available quantification techniques for DFTs have motivated this thesis to explore different possible ways to quantify Pandora temporal fault trees to make it equally applicable for both qualitative and quantitative analysis. Under this motivation, part of the scope of this thesis is to develop methodologies to provide state-space solutions to Pandora TFTs and also to provide ways for probabilistic evaluation of the TFTs of systems having components with both exponentially and non-exponentially distributed data.

1.2.3 Challenges of Uncertainty

One common feature of all the MBSA techniques is that they perform quantitative analysis based on fixed values of failure data of system components, and hence take it as guaranteed that the precise failure data of components are always available. However, in the very early stages of design, sometimes it is necessary to consider failure data of new components which have no available failure data. In some cases the exact choice of component has yet to be made and thus precise failure data could not possibly be known. In these kinds of situations, human judgement by linguistic expressions, such as '*very low, low, high, very high*' can be used to define failure data (Mahmood *et al.*, 2013). Fuzzy logic is a branch of mathematics developed by Zadeh (1965) that deals with linguistic variables and provides an efficient way to handle uncertain data.

As the outcomes of the quantitative analysis are entirely dependent on the accuracy of the data used in the analysis, the optimistic assumptions regarding the availability of precise failure data may produce misleading results or, in the worst case, the quantitative analysis may need to be discontinued due to the unavailability of the failure data. As mentioned earlier, the issue of uncertain quantitative data has been addressed in the quantitative analysis of classical and dynamic fault trees but no effort has been made to address the issue of uncertainty in the quantification process of Pandora temporal fault trees. Now the question is: how can the uncertainty about the component failure data be incorporated in the quantification process of compositional MBSA techniques? This forms another motivation of this thesis.

The answer to the above question requires formulating methodologies which allow quantitative evaluation of system reliability with limited or absent quantitative data about system components. As Pandora is easily integrated with MBSA techniques, the scope of this thesis is to explore different possibilities and develop methodologies to incorporate the issue of uncertainty into the quantification process of Pandora.

1.3 Research Hypothesis and Research Questions

Based on the challenges in the area of MBSA as presented in the previous sections, and in the context of the motivation indicated in the previous section, the hypothesis defended in this thesis is as follows:

It is possible to extend a state-of-the-art method for MBSA with novel concepts that can address challenges in quantitative analysis of dynamic systems using Pandora and the more general challenges of incorporating uncertainty in the field of advanced MBSA.

In order to do that a set of research questions have been developed as follows:

1. How can we model the dynamic behaviour of systems and link it to dynamic analysable models that can be used by compositional MBSA techniques to perform dynamic dependability analysis?

MBSA techniques use formal system models to facilitate automatic system analysis. To aid in the automatic generation of dependability analysis artefacts such as fault trees and FMEAs, system models are annotated with dependability related information. This annotation typically uses Boolean expressions describing a fixed failure behaviour for each system component. However, modern complex systems can operate in multiple modes and the system components can have multiple sets of failure behaviour depending on the mode of system operation, i.e., system components exhibit dynamic behaviour. This makes Boolean expressions alone insufficient to perform dynamic analysis and gives rise to the necessity of mode-based annotation of dynamic components. State machines are good candidates for this and are widely been used in different techniques as a means to describe dynamic system behaviour. Therefore, we believe: a state-transition formalism can be used to capture dynamic failure behaviour and therefore allow compositional MBSA techniques to perform automated qualitative and quantitative dependability analysis of dynamic systems.

2. How can the Pandora temporal fault trees of a dynamic system consisting of components with various failure distributions be probabilistically evaluated using state-space based approaches?

Pandora can be used with MBSA techniques to capture sequence-dependent dynamic behaviour of systems. As Pandora captures the information about the state of a system, standard combinatorial techniques used for probabilistic evaluation of classical fault trees cannot be used to quantify Pandora TFTs. The solution requires generating all possible system states and stochastic transitions between states. The existing analytical method of quantifying Pandora temporal fault trees is applicable only for systems featuring components with exponentially distributed lifetimes. However, components can have failure rates with other distributions, like Lognormal or Weibull. Different methodologies have

been developed to provide state-space solution to DFTs of dynamic systems featuring various failure distributions. As Pandora TFTs are also a DFT-like dynamic extension of classical fault trees, we believe: state-space quantification methods can be used on Pandora temporal fault trees to perform probabilistic dependability evaluation of dynamic systems with various failure rate distributions.

3. How can we incorporate the issue of uncertain component failure data in the quantification process of temporal fault trees and thereby allow probabilistic evaluation of dynamic systems in conditions of uncertainty?

Quantitative dependability analysis of a system is performed based on the failure logic and the probabilistic failure data (e.g. failure rate or failure probability of component) of the system components. The logical expressions showing the possible causes of system failure can be obtained from the qualitative analysis but the failure data is completely dependent on the components themselves. That means the availability of failure data is the deciding factor of the possibility of performing a quantitative analysis and we have seen earlier that there are situations when failure data of some component may not be available. Although there are multiple issues with assuming the certainty of component failure data, these are frequently ignored in MBSA techniques, i.e., they take it as guaranteed that the failure data are always available and accurate. This assumption about the availability of failure data could undermine the applicability of such MBSA techniques for quantitative analysis under uncertainty. In the past, research has been performed to address the issue of uncertainty in the quantification process of classical FTs and DFTs, but none of these efforts were made in the context of MBSA. As Pandora can be integrated with compositional MBSA techniques, we believe: it is possible to incorporate uncertain failure data in the quantification of Pandora TFTs, thus allowing MBSA techniques to perform quantitative analysis in conditions of uncertainty.

1.4 Research Objectives

To investigate the validity of the above research hypothesis, this thesis systematically researches, develops and evaluates the proposed methodologies in five major steps, which form the key objectives of the thesis:

1. Identify the complications faced by the MBSA techniques due to the dynamic behaviour of systems.

To perform dynamic dependability analysis, one needs to understand what the dynamic behaviour of a system really means. To improve the ability of compositional MBSA techniques to perform dynamic dependability analysis, it requires proper understanding of the challenges posed by the dynamic system behaviour and why these challenges complicate

the system analysis process and make the results of analysis inconsistent. This objective aims to describe the issues surrounding dynamic system behaviour in general and then identify different complications it causes in the dependability analysis process. In summary, this objective will identify a set of key challenges of dynamic analysis that the subsequent objectives should overcome.

2. Provide a methodology to represent dynamic system behaviour

This objective is to determine how the components of the system can be annotated with dynamic information. To accomplish this task, architectural models of systems will be examined to see how the static behaviours of the system are annotated using Boolean logic and different state-transition formalism based annotation techniques will be studied to evaluate their structure, strengths and weaknesses to determine their potential use in defining dynamic behaviour of systems. The goal is to develop a methodology to model the dynamic behaviour of systems using state-event automata (e.g. mode or state charts (Harel, 1987)). It will extend the existing Boolean annotation techniques used in compositional MBSA approaches such as HiP-HOPS, thus combining both static and dynamic behaviour of a system in a single annotation. Temporal logic ought to be an option that analysts can choose whenever necessary to define dynamic behaviour, but not something they are forced to use all of the time. This thesis will also investigate the potential complexities that may arise from the mode-based architectural annotation and provide guidelines on potential use of hierarchical system modelling using mode charts to alleviate the complexities.

3. Demonstrate the dynamic system modelling and dependability analysis process in HiP-HOPS

This objective aims to demonstrate the use of the methodology developed under the previous objective for dynamic system modelling in the context of compositional model based safety analysis. Although the developed methodology is tool independent, the HiP-HOPS tool will be used to demonstrate how it works. The reason behind choosing HiP-HOPS is that it is a well-supported, state-of-the-art MBSA tool and it has additional functionality (e.g. automated dependability analysis, architecture optimisation, safety requirements allocation etc.) that could benefit from improved support for dynamic modelling, resulting in a greater contribution to the field of MBSA. In this objective, the use of the developed methodology will be illustrated by applying it to a case study of a dynamic system and evaluating the results.

4. Develop methodologies to provide a state-space solution to Pandora TFTs

Dynamic system behaviour defined using mode charts cannot be directly analysed to obtain useful dependability related information. For meaningful analysis, mode charts are transformed into alternative representations such as Pandora temporal fault trees. To probabilistically evaluate the system reliability from the Pandora TFTs, one needs to identify how to probabilistically quantify all Pandora TFT gates, each minimal cut sequence that can cause system failure, and eventually the top event probability. It is also important to identify critical system components so that analysts can prioritise components according to their contribution to the occurrence of the system failure. Pandora TFTs include both Boolean and temporal gates.

Since the temporal gates capture the order of events occurrence, i.e., state of the system, the quantitative evaluation of each of the Pandora TFT gates requires generating state-space solutions for each of the gates according to their logical behaviour as defined by Pandora. The existing analytical approach to quantify Pandora TFTs is dependent on the exponential distribution of failure data. An objective of this research is therefore to explore different techniques to provide a state-space solution to Pandora TFTs and subsequently develop methodologies to evaluate the reliability of dynamic systems. One aspect of this is to develop an evaluation technique for Pandora TFTs that would be equally applicable to exponentially and non-exponentially distributed data.

5. Develop a methodology to enable uncertain data to be used in the quantification of Pandora TFTs

The probabilistic evaluation of system dependability is performed based on failure data of system components. Irrespective of the methodologies used to perform the quantitative analysis, it is generally assumed that the failure data of system components are always available. However, in the early design phase, failure data of some system components may not be known. To address the issue of uncertain data, another objective of this thesis is to investigate the possibility of incorporating uncertainty in the quantification process of Pandora TFTs. The result would be a methodology to allow analysts to perform quantitative analysis of dynamic systems even with limited or absent probabilistic failure data, thereby providing a way for MBSA techniques to perform dynamic analysis under conditions of uncertainty.

1.5 Assumptions

The following assumptions are made in this thesis.

Non-Repairable Components

It is assumed that the system components are non-repairable. That is, once a component

fails it remains in the failed state forever. In other words, the state of a system component can change from non-failed (false) to failed (true) and not vice versa.

Statistical Independence
System components are considered to be statistically independent, which means that the probability of failure of a component is not dependent on the probability of failure of another component in any way.

1.6 Thesis Structure

The rest of the thesis is arranged as follows:

Chapter 2: Background Study and Literature Review

This chapter defines the context of the research by discussing the dependability analysis process. An overview of key traditional and model-based safety analysis methods is provided. A critical review of the literature highlighting the key principles, strengths and limitations of different safety analysis methods is provided to show the background and motivation for the methodologies developed in this thesis.

Chapter 3: Dynamic Dependability Analysis

This chapter explains dynamic behaviour of systems in general and then identifies and discusses different complications that are caused by the dynamic behaviour in the dependability analysis process for the model based safety analysis approaches. After that, it introduces a potential technique for modelling dynamic system behaviour using mode charts. It shows how modes can be associated with the failure behaviour of components and illustrates the mode based annotation process on a simple example system. It also provides some guidelines on hierarchical system modelling to alleviate the complexities that may arise from the mode based representation and also discusses the application of temporal logic to capture dynamic behaviour.

Having presented this modelling technique, this chapter then illustrates the whole process of dynamic dependability analysis using the HiP-HOPS tool which begins with the component annotation, proceeds with generating temporal fault trees from the synthesis of the mode charts, and ends by analysing the temporal fault trees to produce the minimal cut sequences. Finally, this chapter demonstrates how the new approach can be used in practice to perform dynamic analysis of systems by applying it on an Aircraft Fuel Distribution system.

Chapter 4: Quantitative Dependability Analysis under Uncertainty

This chapter studies the existing quantitative analysis methods used for probabilistic evaluation of Pandora TFTs and makes a new contribution by presenting an approach for performing dynamic analysis under conditions of uncertainty.

Taking the logical specification of the temporal gates as defined by Pandora into account and keeping all the assumptions unchanged, this chapter proposes two new methodologies to provide a state-space solution to Pandora TFTs and also to determine the criticality of system components. The first methodology proposes translating temporal fault trees into Petri Nets. In this methodology, time is considered as continuous and the components are considered to have exponentially distributed failure data. The second methodology is based on Bayesian Networks. This methodology models time in a discrete domain and could work with any type of failure data distribution. The methodologies are illustrated by applying them to a simple example.

This chapter also proposes a methodology based on fuzzy set theory and expert elicitation for quantitative evaluation of Pandora TFTs with uncertain failure data.

Having developed methodologies for the probabilistic evaluation of Pandora TFTs and for the quantitative analysis of Pandora TFTs with uncertain failure data, this chapter then illustrates how they can be used in practice by applying them to a fault tolerant fuel distribution system of a ship. The analyses are explained step-by-step and the results are discussed. The results are compared with the results produced by other existing methodologies, showing how the proposed methodologies can produce important quantitative information about the dependability of a system under uncertainty.

Chapter 5: Discussion and Evaluation

This chapter evaluates the achievements of this thesis against its objectives. It provides a comparative analysis of the proposed techniques to evaluate them. Different issues related to the proposed approaches and their limitations are described. A potential direction for future research is also presented here.

Chapter 6: Conclusion

This final chapter summarises the work presented in this thesis.

References

A list of references used in this thesis.

Chapter 2

Background Study and Literature Review

This chapter clarifies the scope of the research presented in this thesis, and establishes a context for the research by introducing the readers to different approaches and tools used for dependability analysis of system. Firstly, key terms used in dependability analysis process are defined and a brief description of the model based safety assessment process is provided. Secondly, some of the well-established and widely used classical safety analysis techniques like Failure Mode and Effects Analysis and fault tree analysis are described. Thirdly, different compositional failure logic synthesis and analysis approaches like Failure Propagation and Transformation Notation (FPTN), HiP-HOPS, and AADL as well as other related methods are presented. Fourthly, different behavioural fault simulation approaches like FSAP/NuSMV, and Altarica are described. Fifthly, other alternative safety assessment approaches like Bayesian Networks, and Petri Nets are presented. Relative strength and weakness of different methods are emphasised while describing their characteristics and functionality, and thus help define the scope of further research to overcome the limitations of those conventional methods or to combine different methods to benefit from their strong properties. Finally, concluding remarks are made in the last section.

2.1 Reliability Engineering and Model Based Safety Assessment

Dependability is a broad term which includes aspects of safety, reliability, security, availability, and maintainability. This thesis focuses on the safety and reliability as aspects of dependability, and this subsection defines different terms related to these aspects. Before explaining the technical details of system safety, it is important to establish the definitions of some of the key terms used in this thesis. At first, we have to understand what does the word **safety** mean and when we can say something is safe? In general, safety is defined

as freedom from accidents or losses. But there is an argument between people that there is nothing that could be considered as absolute safety, therefore this term is required to be defined in terms of acceptable loss. By considering this criterion, William Lowrance defined safety as "a judgement of the acceptability of risk, and risk, in turn as a measure of the probability of and severity of harm to human health. A thing is safe if its attendant risks are judged to be acceptable" (Lowrance, 1976). The term acceptable itself is ambiguous, because a condition which is acceptable to an employer or for a particular situation may not be acceptable in different situations or to a different employer. The term **risk** is also defined as the hazard level combined with possibility of the hazard leading to an accident and severity of the potential consequences (Leveson, 1995). A **hazard** is a "state or set of conditions of a system (or an object) that, together with other conditions in the environment of the system (or object), will lead inevitably to an accident (loss event)" (Leveson, 1995).

Reliability is a closely related term to safety with subtle differences. Reliability is defined as "the probability that a piece of equipment or component will perform its intended function satisfactorily for a prescribed time and under stipulated environmental conditions" (Leveson, 1995) (with nearly identical definition in (Villemeur, 1991)). Something is considered as reliable if it is unlikely to fail. Unreliability, the complement of reliability, is defined as the probability of failure. **Failure** is "the non-performance or inability of the system or component to perform its intended function for a specified time under specifies environmental conditions" (Leveson, 1995). Some useful parameters related to reliability and unreliability are failure rate, repair rate, Mean Time To Failure (MTTF), Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR). Failure rate is the frequency with which a system or component fails, generally expressed as a number of failures per hour. It is often denoted by λ and has the unit of t^{-1}. The repair rate μ is the probability density (that is, probability per unit time) that the component is repaired at time t given that the component failed at time zero and had been in a failed state (that is, the component is not yet restored to service) to time t. **MTTF** is the average time before the first failure occurs for a repairable component and for a non-repairable component this is the average lifetime of the component. **MTBF** is used for repairable components and it represents the time between failures excluding the down time; i.e., it means up time between failures. **MTTR** is the average time between a failure and a repair; i.e., duration of time between the moment of component failure and the moment it becomes functional after repair.

The aim of reliability engineering is to improve the reliability of a system in order to minimise the risk associated with the system failure or to improve efficiency while reducing the cost. Analysts can discover the flaws of a system through analysis, and therefore can take necessary actions to improve the system design by adjusting to reduce those flaws. There are many different techniques available to perform the safety analyses. Safety analysis techniques like fault tree analysis (FTA) and Failure Modes and Effects

Analysis (FMEA) are well established and widely used during the design phase of safety-critical systems. Usually these types of techniques are manual processes and performed on an informal system model by a single person or a group of persons to fulfil safety requirements of the systems. Although these techniques can produce a great deal of valuable information about the safety and reliability of the system, the overall performance of these techniques largely depend on the skill of the analysts. As these analyses are performed on informal models, it is therefore unlikely that they will be complete, consistent, and error free which make it difficult to reuse that information. Furthermore, manual analyses are usually time consuming and expensive, therefore once performed they are unlikely to be repeated or iterated upon.

Over the past 20 years, research has focused on simplifying the dependability analysis process by automating the synthesis process, which led to a body of work on model-based safety assessment (MBSA) and prediction of dependability. Several approaches to automated safety analysis have emerged, motivated mainly by the increased complexity of systems and increased time and costs associated with the manual analysis. In model-based safety analysis, system designers and safety analysts both use the same system model or somehow related models. As a result, the models become more formal than a separate model for safety analysis. This can allow automating all or some part of the safety analysis process. By automating the safety analysis processes, MBSA can save time and expenses and allow the reusability of the information. More importantly, it allows the analysts to repeat the analyses process as many times as they want to cope with the frequent changes in the model of the systems; it also reduces the probability of introducing errors and omitting important information.

Within model-based safety analysis, two different paradigms of research have been developed (Walker *et al.*, 2008). The first paradigm can be called failure logic synthesis and analysis which focuses on the automatic construction of predictive system failure analyses. The other paradigm can be called behavioural fault simulation which focuses more on behavioural simulation to automatically analyse potential failures in a system. Some of the well-known classical safety analysis techniques and MBSA approaches of each category will be discussed in the next sections.

2.2 Classical Safety Analysis Techniques

This section reviews different classical and well-established safety analysis techniques which include Failure Mode and Effects Analysis (FMEA)/ Failure Mode Effects and Criticality Analysis (FMECA) and Fault Tree Analysis (FTA).

2.2.1 FMEA/FMECA

Failure Mode and Effects Analysis (FMEA) is a well-known inductive safety analysis method. FMEA was developed by reliability engineers in order to predict component

reliability. The FMEA process is described in both academic literature e.g., (Leveson, 1995) and in a number of industrial standards like (IEC, 2006). The FMEA process starts with defining the system, identifying and listing all components of the system with their failure modes, considering all possible modes of operation. Afterwards, the effects of each of the failure modes on the other components of the system as well as on the overall system is determined. Then the severities and probabilities of each failure modes are evaluated. The results are documented in a table with column headings like item, failure mode, and effects.

Failure Modes, Effects, and Criticality Analysis (FMECA) is an extension of FMEA with more detailed analysis of the criticality of the failure modes. Two additional steps are added to the FMEA: (1) identifying failure detection methods and compensating provisions, and (2) identifying corrective actions to eliminate the failure or to reduce the risk (Pukite and Pukite, 1998). Example of an FMECA table is shown in Table 1 (Leveson, 1995). FMEA/ FMECA do not usually provide any systematic ways for identifying failure modes or determining their effects on other components. The following failure conditions must be considered while identifying failure modes and their causes (Pukite and Pukite, 1998):

1. Premature operation.
2. Failure to operate at a prescribed time.
3. Intermittent operation.
4. Failure to cease operation at a prescribed time.
5. Loss of output or failure during operation.
6. Degraded output or operational capability.
7. Other unique failure conditions.

The strength of these techniques is their completeness. However, the processes are very time consuming and if applied to all parts of a complex system, then it can become monotonous and costly. FMEA does not normally consider effects of multiple failures at a single time. It normally considers each failure as an independent event without any relation with other failures except the subsequent effects the failure might produce (Leveson, 1995). But the FMEA used in the AutoSteve tool (Price and Taylor, 2002) can consider the effects of multiple failure modes. As it has to consider all possible combinations of failure, it would lead to a combinatorial explosion. Papadopoulos *et al.* (2004b) have proposed a method to consider the effects of multiple failures and the methodology proposed by Walker *et al.* (2009) can consider the effects of multiple failures as well as their sequential behaviour.

Table 1: A sample FMECA

Item	**Failure Modes**	**Cause of Failure**	**Possible Effects**	**Probability**	**Level**	**Possible Actions to reduce Failure Rate or Effects**
Motor Case	Rupture	Poor workmanship. Defective materials. Damage during transportation. Damage during handling. Over pressurisation.	Destruction of missile.	0.0006	Critical	Close control of manufacturing processes to ensure that workmanship meets prescribed standards. Rigid quality control of basic materials to eliminate defectives. Inspection and pressure testing of completed cases. Provision of suitable packaging to protect motor during transportation.

Among the different model-based safety analysis tools HiP-HOPS (Papadopoulos, 2000) has the capability to automatically generate FMEA from the fault trees. FMEA produced by HiP-HOPS can describe further effects of a failure mode along with the direct effect of that failure mode on the system. Grunske *et al.* (2007) have proposed a method for probabilistic failure modes and effects analysis (pFMEA) using probabilistic fault injection and model checking methods.

2.2.2 Fault Tree Analysis: FTA

Fault tree analysis (FTA) is a well-established and widely used method for evaluating system safety and reliability qualitatively as well as quantitatively (Vesely *et al.*, 2002). After its creation in the 1960s, it has been used in variety of fields, including but not limited to: automotive, aerospace, and nuclear industries (Walker and Papadopoulos, 2009). Fault trees utilise graphical representations based on Boolean logic to show logical connections between different faults and their causes (Vesely *et al.*, 1981). They are a deductive analysis method, which means analysis starts with a system failure known as the 'top event' and works backwards to determine its root causes. From a fault tree, it is possible to understand how combinations of failures of different components or certain environmental circumstances can lead to system failure. The roles of FTA in decision making are (Vesely *et al.*, 2002):

- To understand the logic leading to the top event.
- To prioritise the contributors leading to the top event.
- As a proactive tool to prevent the top event.
- To monitor the performance of the system.
- To minimise and optimise resources.
- To assist the designing of the system.
- As a diagnostic tool to identify and correct causes of the top event.

Steps required for successful FTA and the interrelationship between the steps are shown in Figure 2. The first step is to define the objective of the FTA, i.e., defining the objectives in terms of a failure of the system to be analysed. Just after defining the objective of FTA, the top event of the FT is defined in step 2. After that, the scope of the FTA is defined by indicating which of the component failures and contributions will be considered and which will not be considered. The level of detail to which the failure causes for the top event will be developed is defined is step 4 as the resolution of FTA. In step 5, ground rules for the FTA are defined by declaring the procedure and nomenclature by which events and gates are named. The ground rules are very important in creating an understandable FT.

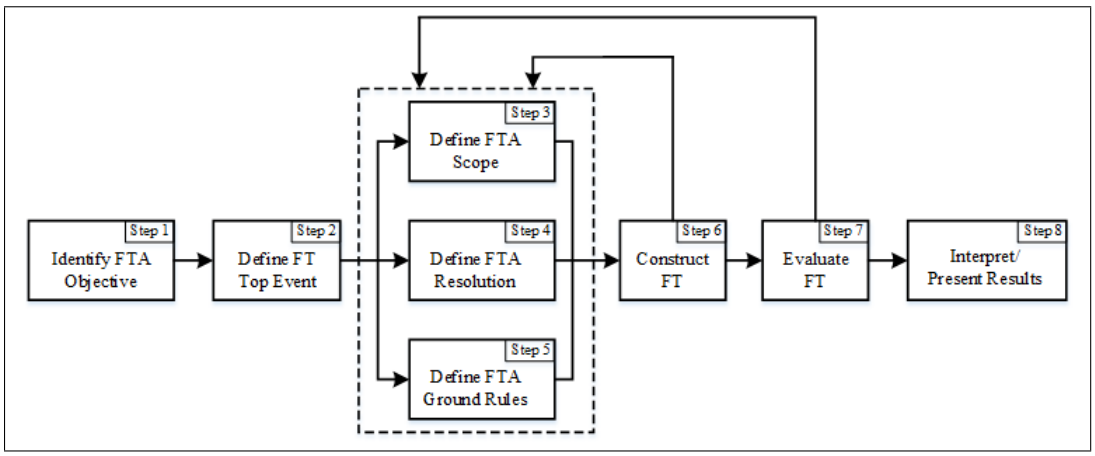


Figure 2: Fault Tree Analysis Steps (Vesely *et al.*, 2002)

The actual construction of a FT is performed in step 6 and the evaluation is done in step 7. Both qualitative and quantitative evaluations could be performed. The qualitative evaluation provides information about the minimal cut sets which are the smallest combinations of events that are necessary and sufficient to cause the top event. The quantitative evaluation usually provides the probability of the top event and importance measure of events based on their contribution to the top event. Finally, at the last step the results are interpreted and presented.

2.2.2.1 Fault Tree Events and Gates Symbols

Fault trees use different symbols to represent different events and gates. Symbols used in classical FTs to represent different events are shown in Figure 3.

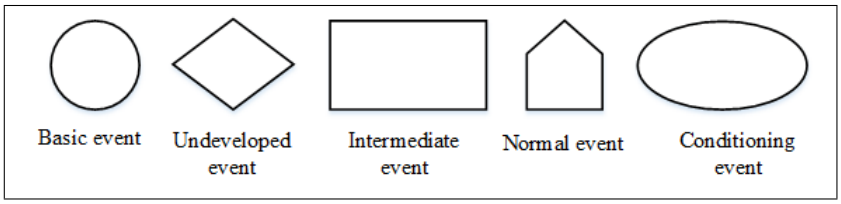


Figure 3: Fault Tree Event Symbols

A basic event is an initiating or basic fault that does not require any further development or expansion and is graphically represented by a circle. Basic events are represented as leaf nodes in the fault tree and they combine together to cause intermediate events. To facilitate quantitative analysis basic events are usually given failure rates and/or repair rates. And in the qualitative analysis, cut sets are the combination of different basic events.

An intermediate event is a fault that is caused by the logical combinations of other events occurring further down the tree. As intermediate events are caused by other events, they are almost always a type of logical gate. An undeveloped event is an event whose contributions are not considered in the analysis, either because it is considered as unnecessary, or because insufficient information is available. It is graphically represented by a

diamond. A conditioning event does not necessarily represent a fault, it serves as a special condition or constraint for certain types of gates. An ellipse is used to represent a conditioning event. A normal event does not represent any fault and it is part of the nominal behaviour of the system. Normal events are represented by a house symbol.

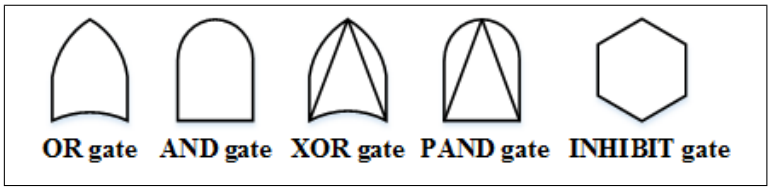


Figure 4: Fault Tree Logic Gate Symbols

Symbols used in classical FTs to represent different logic gates are shown in Figure 4. The OR gate is used to show a scenario when the output event occurs if at least one of the input events occur. There is no restriction on the number input events to an OR gate. Inputs to an OR gate are often restatements of the output i.e., an OR gate does not necessarily represent a causal relationship between its inputs and outputs. In this thesis, the symbol '$\vee$' is used to represent an OR gate in text or in logical expressions. The output of an AND gate is true if all of its input events are true. For example, a fire detection system can fail if both smoke detector unit and heat detector unit fail but not by the failure of just one unit. Similar to the OR gate there may be any number of input events to an AND gate but in contrast to the OR gate, the AND gate usually represents a causal relationship between its inputs and outputs. The symbol '$\wedge$' is used to represent an AND gate in text or in logical expressions in this thesis. The XOR gate is true if one and only one of its input events is true. This gate is a special case of the OR gate and in most fault tree analysis it is considered as a two-input gate where the output is true if only one of the inputs is true but not two[1]. The symbol '$\oplus$' is used to represent an XOR gate in text or in logical expressions. The PAND gate is a special case of the AND gate and detail about the PAND gate will be provided in subsection 2.2.2.5. The INHIBIT gate is a special case of the AND gate and it produces an output when its only input event is true in the presence of a conditioning event. An example of a typical fault tree is shown in Figure 5.

2.2.2.2 Qualitative and Quantitative Analysis of Fault Trees

After construction of a fault tree, qualitative analysis is performed using Boolean logic by reducing it to minimal cut sets (MCSs), which are a disjoint sum of products consisting of the smallest combinations of failure events that are necessary and sufficient to cause the top event. The order of a minimal cut set defines the number of basic events that contribute to that minimal cut set. A 1^{st} order MCS consists of a single basic event,

[1]In digital logic design, when a XOR gate has more than two inputs then the output becomes true when odd number of inputs become true.

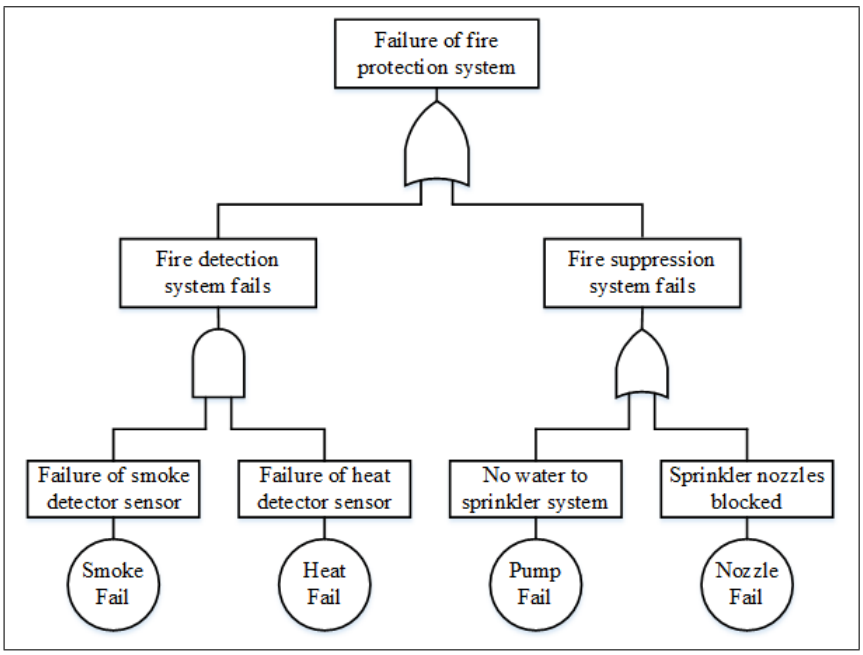


Figure 5: Example of a Fault Tree (Andrews, 1998)

i.e., a single failure event alone can cause the top event to occur. Therefore, this single component becomes a candidate for upgrade or to replicate. On the other hand, a 4^{th} order MCS contains four basic events. The lower the order of a MCS the higher the importance of that MCS is. There are many fault tree algorithms available to perform qualitative, quantitative, and other different analysis. Walker (2009) has described some of the fault tree analysis algorithms like MOCUS — Method of Obtaining Cut Sets (Fussel and Vesely, 1972), ELRAFT — Efficient Logic Reduction Analysis of Fault Trees (Semanderes, 1971), Linear Time Modularisation Algorithm (Dutuit and Rauzy, 1996), and BDD — Binary Decision Diagram (Sinnamon and Andrews, 1997).

Quantitative analysis of a fault tree, which follows qualitative analysis, can help to estimate the probability of the top event occurring from the given failure rates of basic failure modes of the system (Vesely *et al.*, 1981). In the quantification process, the basic events are assumed to be statistically independent. Usually, in fault tree analysis, as the top event is represented as the disjoint sum of the MCS, an approximate value of the probability of the top event can be determined by calculating the probability of each MCS and then adding them together, given that the probability of MCSs are small. In the Fault Tree Handbook (Vesely *et al.*, 2002), this approximation is termed as "rare event approximation" and it is also stated that if the basic events probabilities are below 0.1 then this approximation are typically sufficiently accurate. In addition to this approximation, depending on the applications, different kinds of probabilities like time-dependent probabilities could be calculated provided that the proper failure distributions of the components/events are available.

Although the primary focus of the quantitative analysis of a fault tree is to determine the top event probability, it is possible to determine the probability of any intermediate events as well as the basic events. Dominance of the minimal cut sets could be deter-

mined based on the significance of their contribution to the top event. The cut set which contributes the most to the top event is considered as the most dominant. In addition to determining dominant MCS, importance of basic events could also be obtained in the similar way.

To be able to perform quantitative analysis to get top event probability, the basic events are usually given one of the following types of data (Vesely *et al.*, 2002):

1. a component failure probability in some time interval,
2. an event occurrence probability in some time interval,
3. a component unavailability, and
4. a pure event probability.

If the exponentially distributed failure rate (λ) of a component C and the mission time is given, then the failure probability of the component is calculated as:

$$Pr\{C\}(t) = 1 - e^{-\lambda t} \tag{1}$$

where $Pr\{C\}$ is the failure probability of the component, also known as the unreliability of the component and t is the mission time.
As the minimal cut sets are presented as the intersection of the basic events, the probability of a MCS is obtained by simply multiplying the probabilities of the basic events as follows:

$$Pr\{MCS_i\} = \prod_{j=1}^{n} Pr\{BE_j\} \tag{2}$$

where $Pr\{MCS_i\}$ is the probability of the minimal cut set i and $Pr\{BE_j\}$ is the probability of the basic event j in MCS_i.
And, since the top event is the union of the minimal cut sets, hence, the top event probability could be calculated as (Esary and Proschan, 1963):

$$Pr\{top\ event\} = 1 - \prod_{i=1}^{n} \left(1 - Pr\{MCS_i\}\right) \tag{3}$$

where $Pr\{top\ event\}$ is the probability of the top event and $Pr\{MCS_i\}$ is the probability of the minimal cut set i.

The outcomes of the quantitative evaluation of a fault tree largely depend on the precision of the data used during the analysis. Imprecise or uncertain data could make the results of a quantitative analysis questionable. Though FTA is popular and widely used, application of traditional FTA has some limitations, e.g., in expressing time- or sequence-dependent dynamic behaviour (Dugan *et al.*, 1990; Bruns and Anderson, 1993; Walker

and Papadopoulos, 2009); or when handling uncertainties, allowing use of linguistic variables and integrating human error in failure logic (Mahmood *et al.*, 2013; Tanaka *et al.*, 1983). These are well known limitations of FTA and by this time conventional FTA has gone through different modifications to overcome them.

To capture sequence-dependant behaviour, classical FTs are extended to introduce Dynamic Fault Trees (DFTs) and Pandora Temporal Fault Trees. The basic description of DFT is presented in subsection 2.2.2.4 and the description of the Pandora Temporal Fault Tree is presented in subsection 2.2.2.5. To handle uncertainties during quantitative analysis, classical fault trees are extended using fuzzy set theory, known as Fuzzy Fault Trees. Some preliminaries on fuzzy fault trees are provided in the next subsection.

2.2.2.3 Fuzzy Extension of FTA: FFTA

During quantitative analysis, in conventional FTA, failure rates of components are typically considered to be constant (Yuhua and Datao, 2005). But for many complex systems, it is often very difficult to estimate a precise failure rate due to lack of knowledge, scarcity of statistical data, ambiguous component behaviour, and operating environment of the system (Liang and Wang, 1993; Singer, 1990). This situation is especially relevant in the early design stages when system analysts may have to consider failure rates of completely new or undetermined components which have no available quantitative failure data, and thus precise failure rates could not possibly be known. As Fuzzy Logic is a branch of mathematics that deals with linguistic variables and provides an efficient way to draw conclusions from imprecise and vague information, in order to allow the conventional FTA to capture uncertainty, a different quantification method of FTA based on fuzzy set theory have been proposed by different researchers e.g., (Gmytrasiewicz *et al.*, 1990; Lin and Wang, 1997; Suresh *et al.*, 1996; Tyagi *et al.*, 2011; Wang *et al.*, 2013; Ferdous *et al.*, 2009; Tanaka *et al.*, 1983).

The basic idea of fuzzy FTA is to use fuzzy representations of failure rates or failure probability of components instead of the crisp representations used in classical FTA. A fuzzy number $\boldsymbol{A}$ can be thought of as a set of real numbers where each possible value has a weight between 0 and 1. This weight is referred to as the degree of membership defined by a membership function. Among different forms of fuzzy numbers, triangular fuzzy number (TFN) and trapezoidal fuzzy number (TZFN) are widely used in reliability analysis. A trapezoidal fuzzy number $A = \{a_1, a_2, a_3, a_4\}$ shown in Figure 6 is defined by the membership function as:

$$\mu_A(x) = \begin{cases} \dfrac{x-a_1}{a_2-a_1}, & for\ a_1 < x < a_2, \\ 1, & for\ a_2 \leq x \leq a_3, \\ \dfrac{x-a_4}{a_3-a_4}, & for\ a_3 < x < a_4, \\ 0, & otherwise. \end{cases} \tag{4}$$

where $a_1 \leq a_2 \leq a_3 \leq a_4$.

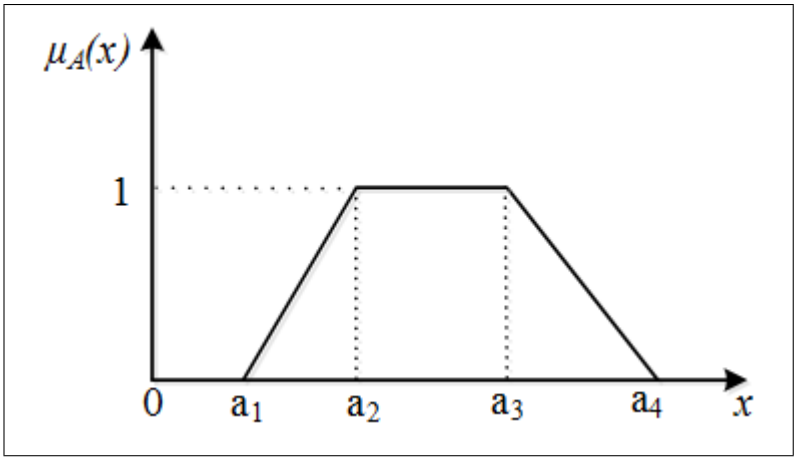


Figure 6: Graphical representation of Trapezoidal fuzzy number

After getting the MCSs from qualitative analysis, the next step is to define or represent the failure rates of all basic components/events as fuzzy numbers. There is no restriction on the format of the fuzzy numbers to be used to represent the failure rates. However, most researchers used either triangular or trapezoidal forms of the fuzzy numbers to represent failure rates of basic events. After defining fuzzy failure rates of all basic events, fuzzy operators for the fault tree's gates are defined and subsequently, all the MCSs are quantified using the fuzzy failure rates of basic events. These values are used to obtain the top event probability. As the fuzzy numbers are used in the quantification process, the top event probability is obtained as a fuzzy number. The idea of fuzzy fault tree analysis is illustrated by an example below.
Let us consider that the failure rates of basic events are defined as triangular fuzzy numbers as:

$$\lambda\left(BE_i\right) = \{\lambda_{i1},\ \lambda_{i2},\ \lambda_{i3}\},\ for\ i = 1,2,\cdots,n \tag{5}$$

where n is the total number of basic events, $\lambda\left(BE_i\right)$ is the fuzzy failure rate of the basic event i and $\lambda_{i1},\ \lambda_{i2},\ and\ \lambda_{i3}$ are the elements of that fuzzy failure rate. By considering mission time as t, the fuzzy probabilities of the basic events are defined using equation (1) as:

$$Pr\{BE_i\}\left(t\right) = \left\{a_i\left(t\right),\ b_i\left(t\right),\ c_i\left(t\right)\right\} = \left\{1-e^{-\lambda_{i1}t},\ 1-e^{-\lambda_{i2}t},\ 1-e^{-\lambda_{i3}t}\right\} \tag{6}$$

As the MCSs are represented as the intersections (AND) of different basic events therefore

from the above fuzzy probability of the basic events and using equation (2) the fuzzy operator of the AND gate or the fuzzy probabilities of MCSs are obtained as:

$$P_{AND} = \prod_{i=1}^{n} Pr\{BE_i\}(t) = \left\{ \prod_{i=1}^{n} a_i(t),\ \prod_{i=1}^{n} b_i(t),\ \prod_{i=1}^{n} c_i(t) \right\} \tag{7}$$

And, finally, if the failure probability of each MCS i is presented by a fuzzy number as $Pr\{MCS_i\}(t) = \{x_i(t),\ y_i(t),\ z_i(t)\}$, then the probability of the top event is obtained using equation (3) as:

$$Pr\{top\ event\} = \left\{ 1 - \prod_{i=1}^{n} (1 - x_i(t)),\ 1 - \prod_{i=1}^{n} (1 - y_i(t)),\ 1 - \prod_{i=1}^{n} (1 - z_i(t)) \right\} \tag{8}$$

For illustration, let us consider the simple fault tree of Figure 7. The logical expression for the top event can be written as:

$$Top\ Event = A_1 \vee A_2 = (X_1 \wedge X_2) \vee (X_3 \wedge X_4)$$

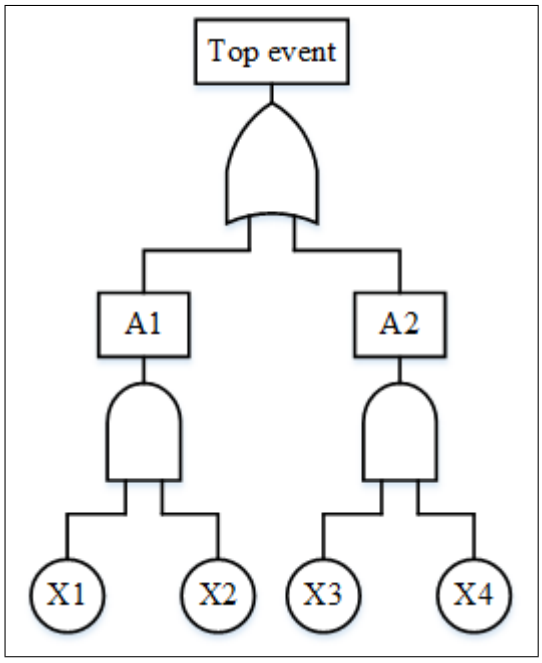


Figure 7: Fault Tree to illustrate the use of Fuzzy Set theory in FTA

Fuzzy failure rates (triangular representation) of basic events and fuzzy probabilities of the basic events considering mission time $t = 1000$ hours are shown in Table 2. Using equation (7) and data from Table 2, the fuzzy probabilities of two MCSs can be obtained as:

$$Pr\{A_1\}(t) = \{0.0002,\ 0.0008,\ 0.0015\} \text{ and } Pr\{A_2\}(t) = \{0.0009,\ 0.0020,\ 0.0030\}$$

Table 2: Fuzzy Failure Rates and Failure Probabilities of Basic Events of Fault Tree of Figure 7

Event Id	**Fuzzy Failure Rates**			**Fuzzy failure Probabilities**		
	λ_{i1}	λ_{i2}	λ_{i3}	$\mathbf{a_i(t)}$	$\mathbf{b_i(t)}$	$\mathbf{c_i(t)}$
X_1	1.005E-5	2.020E-5	3.046E-5	0.01	0.02	0.03
X_2	2.020E-5	4.082E-5	5.129E-5	0.02	0.04	0.05
X_3	3.046E-5	5.129E-5	6.188E-5	0.03	0.05	0.06
X_4	3.046E-5	4.082E-5	5.129E-5	0.03	0.04	0.05

Now using equation (8) and fuzzy probabilities of the MCSs, the fuzzy top event probability is:

$$Pr\{top\ event\} = (0.0011,\ 0.0028,\ 0.0045)$$

The above result implies that the failure probability of the top event is between 0.0011 and 0.0045, and the most possible value of the top event probability is 0.0028. Some researchers like Ferdous *et al.* (2009) have also mapped the fuzzy top event probabilities into a single value. In addition to evaluating fuzzy top event probabilities, most of the researchers those who had used fuzzy fault tree analysis for reliability analysis had also evaluated the fuzzy cut set importance and basic events importance as well.

2.2.2.4 Dynamic Fault Trees: DFTs

As mentioned earlier, classical fault tree analysis is not capable of analysing the failure behaviour of a system where the outcome of the analysis depends not only the combinations of the failures of the components but also on the order in which the components fail. In order to make classical fault tree analysis capable of capturing sequence-dependent dynamic behaviour, a number of extensions like Dynamic Fault Trees (DFTs) (Dugan *et al.*, 1992), Temporal Fault Trees (Palshikar, 2002), State/Event Fault Trees (SEFTs) (Kaiser *et al.*, 2007), and Pandora Temporal Fault Trees (Walker, 2009) have been proposed. The detailed description of all the proposed dynamic extensions of fault trees is beyond the scope of this thesis. However, Dugan's Dynamic Fault Tree is briefly described in this section, while SEFTs and Pandora Temporal Fault Trees will be described in section 2.3.4 and 2.2.2.5 respectively.

Of all the dynamic extensions of fault trees, probably the most prominent is the Dynamic Fault Tree (Dugan *et al.*, 1993; Manian *et al.*, 1998). Two special gates, namely Functional Dependency (FDEP) gate and SPARE gate, are introduced as part of the Dynamic Fault Tree (DFT) to represent temporal behaviour of the systems.

The Functional dependency (FDEP) gate (see Figure 8) helps to design a scenario when the operations of some components of a system are dependent on the operation of another component of the system. For example, when many components of a system

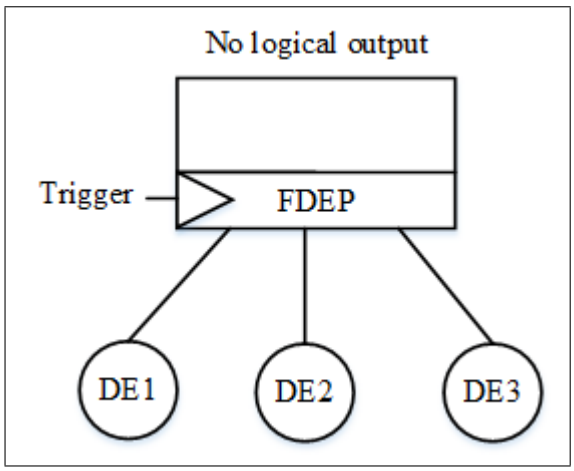


Figure 8: Functional Dependency (FDEP) Gate

receive power from a single source of supply, then failure of the power supply would cause all the dependent components to fail. In a FDEP gate there is only one trigger input which could be a basic event or an intermediate event. However, there could one or more dependent basic events in a FDEP gate. As the dependent basic events are functionally dependent on the trigger event, the occurrence of the trigger event forces the dependent events to occur. By contrast, failure of any number of the dependent events will not affect the trigger event, at the same time dependent events themselves have no effect on each other. The FDEP gate is particularly useful for modelling networked systems, where communication between connected components takes place through a common network element, and failure of the common element isolates other connected components. This type of gate can also model interdependencies, which would otherwise introduce loops in the fault trees.

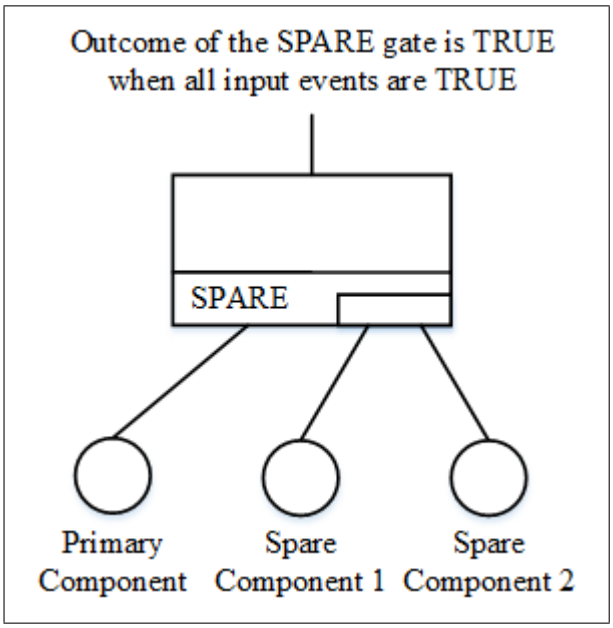


Figure 9: Spare Gate

A SPARE gate is shown in Figure 9. All the inputs of the SPARE gate are basic

events, one of them acts as a primary component (left most input) and the others are the spare components. The SPARE gate designs a scenario where the spare components are activated in a sequence, i.e., if there are two spare components then the first spare will be activated in case the primary fails; if the first spare fails then the second one will be activated. The outcome of the SPARE gate becomes true if all the input events are true. A SPARE gate could model three types of spares: Cold Spares, Warm Spares, and Hot Spares. The failure rate of each of the spare components is affected by the mode they are in and this effect is modelled by a dormancy factor. In the cold spare mode the spare components are deactivated until they are required therefore the dormancy is closer to zero. In contrast, in the hot spare mode, spare components are always active but serve their functionality when the primary fails; as a result the failure rate of a spare component is the same as an active component even if it is not in service, and therefore a spare component has a dormancy factor close to one. In warm spare mode, the spare components are neither on nor off, instead they are kept in-between these two states, i.e., components are kept in a reduced readiness state until required. The dormancy factor of a component in warm spare mode is considered somewhere in-between the dormancy factor of cold and hot spare modes (e.g., 0.5).

Usually, Markov chains are employed to solve DFTs (Dugan *et al.*, 1992) and quantitative analysis of DFTs are performed based on exponential-like distributions of components failure behaviour. As Markov chains are constructed considering all possible combinations of components, the state space of the Markov chain increases nearly exponentially with respect to the components. For this reason, the Markov chain for a large DFT with a significant number of dynamic components will be difficult to create and analyse. Another issue with Markov chains is that they can only quantify the DFT of a system if and only if the components of the system have exponentially distributed failure rates. Alternatives have also been proposed, such as an algebraic framework to model dynamic gates of DFTs; this allows qualitative (Merle *et al.*, 2011) and quantitative (Merle *et al.*, 2014) analysis of DFTs. Moreover, a considerable number of researchers like Malhotra and Trivedi (1995), Hura and Atwood (1988), Bobbio *et al.* (1999), Helmer *et al.* (2007), and Reza *et al.* (2009) have used Petri Nets to solve classical fault trees, however, to the knowledge of the author, fewer efforts (e.g., Codetta-Raiteri (2005); Zhang *et al.* (2009)) have been made to solve DFT using Petri Nets. But uses of Bayesian Networks for solving DFTs are increasing e.g., (Boudali and Dugan, 2005; Marquez *et al.*, 2008; Montani *et al.*, 2008). Further details of Markov chains, Bayesian Networks, and Petri Nets based reliability engineering approaches are presented in section 2.5.1, 2.5.2, and 2.5.3 respectively.

2.2.2.5 Pandora Temporal Fault Trees

Pandora is an extension of classical fault trees, which makes conventional fault trees capable of dynamic analysis (Walker *et al.*, 2007; Walker, 2009). Pandora augments fault

trees with new temporal gates and provides temporal laws to allow qualitative analysis, and thus overcome the limitations of FTA in expressing sequence-dependent behaviour. This technique can be used to determine minimal cut sequences (MCSQs) of temporal fault trees, which are the smallest sequences of events that are necessary to cause the top events, analogous to minimal cut sets (MCSs) of conventional fault trees. The basis of Pandora is the redefinition of the long-established Priority-AND (PAND) gate (Fussell *et al.*, 1976).

Pandora assumes that the occurrence of the events are instantaneous, i.e., go from 'non-fail' to 'fail' with no delay, and that they are persistent i.e., once occurred, they remain in a 'true' state indefinitely. Given this, there are three possible temporal relations between two events X and Y:

- *before* – X occurs first, Y occurs second
- *after* – Y occurs first, X occurs second
- *simultaneous* – X and Y occur at the same time

To represent these three temporal relations between events, i.e., to capture the sequence between the occurrence of events, Pandora introduces three temporal gates: Priority-AND (PAND), Priority-OR (POR), and Simultaneous-AND (SAND).

Priority-AND (PAND)

As mentioned earlier, the PAND gate is not a new gate. It has been used in FTA as far back as the 1970s and also features in the DFT methodology. Nevertheless, it was never properly defined for qualitative analysis, resulting in ambiguous behaviour. Therefore, to avoid ambiguity, in Pandora, the PAND gate is defined as being true if the following conditions are true:

- All input events occur
- Input events occur from left to right
- No input events occur simultaneously

In Pandora, the symbol'$<$' is used to represent the PAND gate, however, to avoid confusion between a PAND gate and the less than sign, this thesis uses '$\lhd$' as the symbol for the PAND gate. As a result, now $X \lhd Y$ means X PAND Y, where X and Y are both failure events. The symbol of the Pandora's PAND gate with n inputs is shown in Figure 10.

Priority-OR (POR)

Similar to the PAND gate, the Priority-OR (POR) gate also defines a sequence, but it specifies an ordered disjunction rather than an ordered conjunction. It is used to indicate that one input event has priority and must occur first for the POR to be true, but does not

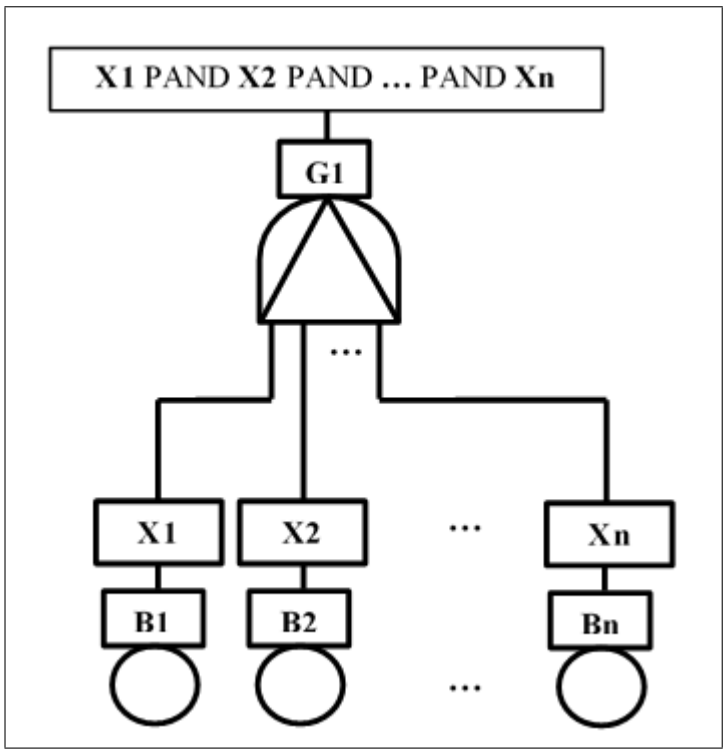


Figure 10: Pandora's PAND gate with n inputs $(n \geq 2)$

require all other input events to occur as well. The POR can therefore be used to represent trigger conditions where the occurrence of the priority event means that subsequent events may have no effect. The POR is true only if:

- Its priority event (left-most) occurs
- No other input event occurs before or at the same time as the priority input event

In Pandora, the symbol '|' is used to represent the POR gate in logical expressions, but this symbol is often confused with the conditional probability symbol. To avoid this confusion, in this thesis, the symbol '⟅' is used to represent the POR gate, thus $X \wr Y$ means (X POR Y) and the graphical symbol of the Pandora's POR gate is shown in Figure 11.

Simultaneous-AND (SAND)

The Simultaneous-AND or SAND gate is used to define situations where an outcome is only triggered if two or more events occur simultaneously. For example, this can happen because of a common cause, or because the events have a different effect if they occur approximately simultaneously as opposed to in a sequence. It is true only if:

- All input events occur
- All the events occur at the same time

The symbol '&' is used to represent the SAND gate in logical expressions and the fault tree symbol of the SAND gate is shown Figure 12.

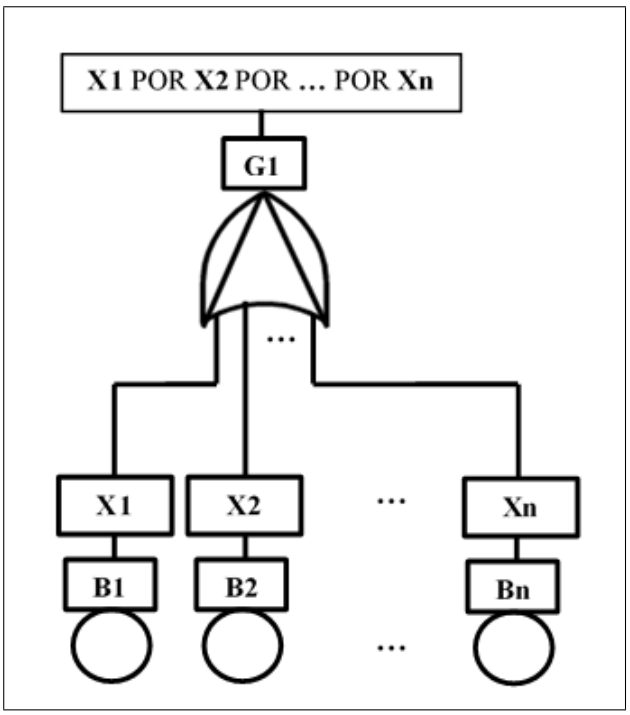


Figure 11: Pandora's POR gate with n inputs $(n \geq 2)$

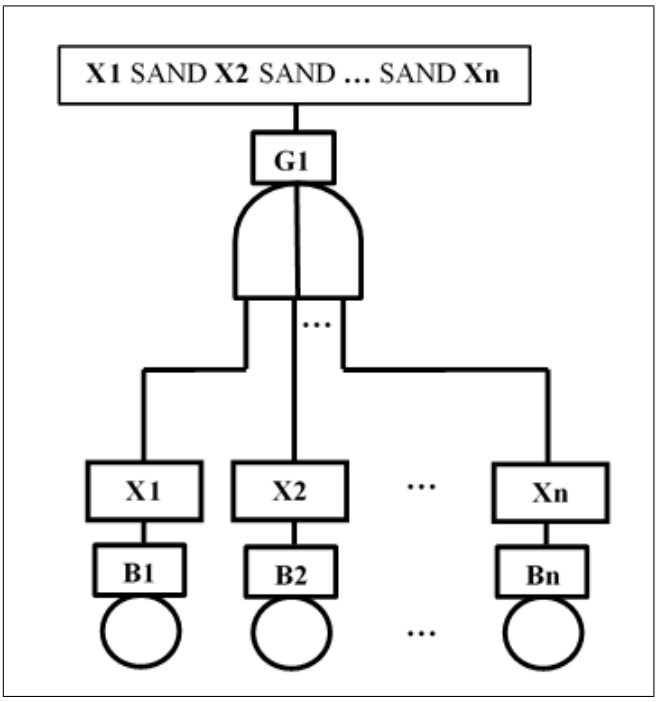


Figure 12: Pandora's SAND gate with n inputs $(n \geq 2)$

These three temporal gates are integrated with other Boolean gates (AND, OR) to create the Pandora Temporal Fault Tree. In Pandora's logical expressions, the SAND gate has the highest priority, then the PAND, POR, AND, and OR. Therefore $P \vee Q \;\&\; R \lhd S \wr T$ is equivalent to $P \vee (((Q \;\&\; R) \lhd S) \wr T)$. As well as the three temporal gates, Pandora also defines a set of temporal laws that describe the behaviour of the gates and how they relate to each other and to the standard Boolean AND and OR gates. These laws include the temporal versions of the Distributive, Associative, Absorption laws, and other fully

temporal laws (Walker, 2009). As temporal expressions are usually complex when compared to Boolean expressions, temporal laws can be used to minimise the expressions and therefore reduce complexity (Walker and Papadopoulos, 2009). Most important of the Pandora laws are the Completion laws (Walker, 2009), which relate the temporal gates to the Boolean gates:

- Conjunctive Completion Law: $X \wedge Y \quad \Leftrightarrow \quad (X \lhd Y) \vee (X \ \& \ Y) \vee (Y \lhd X)$
- Disjunctive Completion Law: $X \vee Y \quad \Leftrightarrow \quad (X \rhd\!\lhd Y) \vee (X \ \& \ Y) \vee (Y \rhd\!\lhd X)$
- Reductive Completion Law: $X \quad \Leftrightarrow \quad (Y \lhd X) \vee (X \ \& \ Y) \vee (X \rhd\!\lhd Y)$

As mentioned earlier, in Pandora, there are only three possible temporal relations between any two events, therefore if both the events occur then one and only one of these relations must be true. The Conjunctive Completion law represent this scenario, i.e., if X and Y both occur, then either X occurred before Y, or they occurred simultaneously, or Y occurred before X. The Disjunctive Completion law represents a scenario when at least one of the two events occurs, then either only one occurred on its own or before the other, or both occurred simultaneously. The Reductive Completion Law shows how an event (in this case Y) can be totally redundant if it does not matter when it occurs in relation to another event.

Table 3: Temporal Truth Table for all gates in Pandora

X	Y	X OR Y	X AND Y	X POR Y	X PAND Y	X SAND Y
0	0	0	0	0	0	0
0	1	1	0	0	0	0
1	0	1	0	1	0	0
1	1	1	1	0	0	1
1	2	1	2	1	2	0
2	1	1	2	0	0	0
2	2	2	2	0	0	2

All the temporal laws can be proved by means of *temporal truth tables* (Walker, 2009). Temporal truth tables are extensions of Boolean truth tables with information about the order in which events occurred. Pandora defines the order of events occurring by means of sequence values. In Pandora, "the exact time at which an event occurs is not important — the only thing that matters is when it occurs relative to the other events, i.e., which comes first, which comes second, which comes last etc." (Walker, 2009). Similarly, it is not necessary to know exactly how long the interval between these events last. Therefore, sequence values are the abstractions of the time at which an event occurs. Sequence values differ from the 'true' and 'false' values of Boolean logic in that they not only express true or false but also the relative order between events. For example, sequence value 0 means the event has not occurred but if an event has occurred then it will have any sequence

value greater than 0 to indicate when it occurs relative to other events under consideration. For example, sequence value 1 means the event occurs first, 2 means second, and so on, i.e., the higher the sequence value the later the event occurs. The temporal truth table for all the gates in Pandora using the concept of sequence value is shown in Table 3 (Walker, 2009).

Quantitative Evaluation of Pandora Temporal Gates

The primary intention behind creating Pandora was to allow dynamic qualitative analysis of systems. However, in terms of quantitative analysis, techniques are also available for probabilistic evaluation. Quantitative solutions to the PAND gate include derivations from calculus (algebraic solution) (Fussell *et al.*, 1976; Merle *et al.*, 2011; Yuge and Yanagi, 2008; Merle *et al.*, 2010); Markov Chain based solutions (Boudali *et al.*, 2007); Bayesian Network based solutions (Boudali and Dugan, 2005; Marquez *et al.*, 2008; Montani *et al.*, 2008, 2006); and Petri Net based solutions (Codetta-Raiteri, 2005; Zhang *et al.*, 2009). As the DFT's PAND gate is not always considered to be the same as the Pandora's PAND gate, not all the solutions will be applicable for Pandora's PAND gate. However, the solution proposed by Merle *et al.* (2010) shown in equation (9) is used as an analytical solution to Pandora's PAND gate.

Recently some analytical approaches have been introduced by Edifor *et al.* (2012, 2013) to quantify Pandora's POR and SAND gates, and the solutions are shown in equations (10) and (11) respectively. Most recently Pandora's temporal gates are quantified by Edifor *et al.* (2014) using Monte-Carlo simulation. But neither the analytical nor the simulation based approaches for evaluating Pandora temporal fault trees consider the inclusion of a degree of uncertainty in the failure rates of the basic events, although a significant amount of research has been done to consider uncertainty in the failure rates of the basic events in the classical fault tree analysis. Moreover, till now no efforts has been made to find Bayesian Network or Petri Net based solutions to Pandora temporal fault trees.

$$Pr\{E_1 \lhd E_2 \lhd E_3 \lhd \ \cdots \ \lhd E_{n-1} \lhd E_n\}(t) = \prod_{i=1}^{N} \lambda_i \sum_{k=0}^{N} \left[\frac{e^{(u_k t)}}{\prod_{\substack{j=0 \\ j \neq k}}^{N} (u_k - u_j)} \right] \tag{9}$$

where $u_0 = 0$ and $u_m = -\sum_{j=1}^{m} \lambda_j \; for \; m > 0$.

$$Pr\{E_1 \wr E_2 \wr E_3 \wr \cdots \wr E_{n-1} \wr E_n\}(t) = \frac{\lambda_1 \left(1 - \left(e^{-(\Sigma_{i=1}^{N} \lambda_i)t}\right)\right)}{\sum_{i=1}^{N} \lambda_i} \tag{10}$$

$$Pr\{E_1 \&_d E_2 \&_d E_3 \&_d \cdots \&_d E_{n-1} \&_d E_n\}\{0, t_0, t_1\} = \sum_{i=1}^{N} \left(F(X_i)\{0, t_0\} * \left(\prod_{\substack{j=1 \\ j \neq i}}^{N} F(X_i)\{t_0, t_1\} \right) \right) \tag{11}$$

2.3 Failure Logic Synthesis and Analysis Approaches to MBSA

In failure logic synthesis and analysis, system-level models of failure propagation are generated from component-level failure behaviour models using a process of composition. As the failure behaviour of the system components is modelled in a compositional way, it is much easier to understand the effects of changing one component or subsystem on the rest of the system. Techniques that follow the compositional approach include: Component Fault Trees (CFTs) (Kaiser *et al.*, 2003); State-Event Fault Trees (SEFTs) (Grunske *et al.*, 2005; Kaiser *et al.*, 2007), Failure Propagation and Transformation Notation (FPTN) (Fenelon and McDermid, 1993), Failure Propagation and Transformation Calculus (FPTC) (Wallace, 2005) and HiP-HOPS (Papadopoulos, 2000). These different techniques will be discussed in the following subsections. As the Error Model Annex of the AADL also follows the compositional approach and is therefore also discussed under this section.

2.3.1 Failure Propagation and Transformation Notation

Failure Propagation and Transformation Notation (FPTN), which overcomes many of the limitations of FTA and FMECA, is the first modular and graphical method to specify failure behaviour of systems with complex architectures. FPTN was created to provide a simple and clean notation to reflect the way in which failures within the system interact along with the system architecture (Fenelon and McDermid, 1993). The basic unit of the FPTN is a "module" and usually is represented by a simple box with a set of input (incoming) and output (outgoing) failure modes. These inputs and outputs are connected to other FPTN modules and a module can contain a number of sub-modules to form a hierarchical structure. Failures can be propagated directly through one FPTN module to another or can be transformed from one type to another. These failures modes could be classified into a number of broad categories, like (Fenelon and McDermid, 1993):

- Timing Failure
  - too early (te), too late (tl) etc.
- Value failures (v)
- Commission (c)
- Omission (o)

Example of an FPTN module is shown in Figure 13 (firstly used by Grunske and Han (2008)). Inside the box, representing a module, there are some standardised sections, each of which consists of some standard attributes like the header section consists of

modules name, and its criticality etc. The most important part is the module specification section (second section) which describes the relationship between input and output failure modes by defining the failure propagations, transformations, generations and detections. This section is important because it defines how the module is affected by the other modules or environment and how other modules or environment are likely to be affected by this module. Each output failure mode of a FPTN module is specified by a single Boolean equation as a sum-of-products of input failure modes (like minimal cut sets of FTA). Therefore a FPTN module could be considered as a forest of trees. In the example FPTN module, one of the output failure modes, *Sprinkling: o* is specified as *Sprinkling: o = Smoke_Detected:tl + Smoke_Detected: o*; i.e., the omission of Sprinkler output could be caused if *Smoke_Detected* input is too late (*Smoke_Detected:tl*) or omission of *Smoke_Detected* input (*Smoke_Detected: o*) occurs.

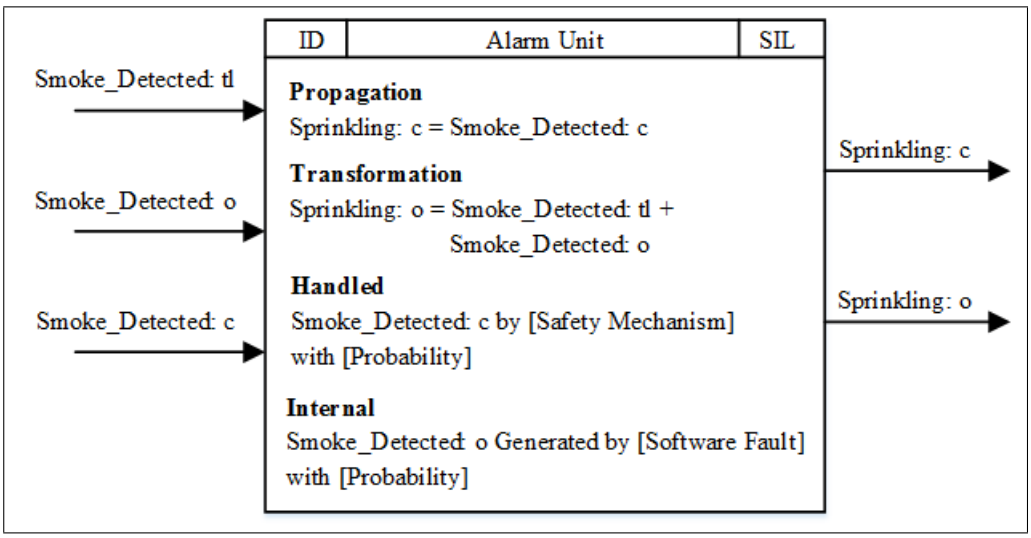


Figure 13: Example FPTN Module

Other than propagating and transforming failures modes from one form to another, a FPTN component can also generate failure modes (internal failure modes) or can handle some of the existing failure modes. These capabilities make FPTN usable as both an inductive (creating FMECA) and a deductive (creating FTA) analysis method. Usually, to evaluate FPTN modules, a fault trees is created for each of the outgoing failure modes. As FPTN are created alongside the design of the system, therefore, if the system model changes then the failure model also change. Potential flaws and problems can be identified from the analysis of FPTN modules and these flaws could be rectified in the subsequent design iterations.

Although FPTN provides systematic and modular notations for representing failure behaviour of systems, in its classical form it can only perform static analysis, not dynamic. However, recently, Niu *et al.* (2011) have extended the classical FPTN notations with temporal information, thus making it capable of performing dynamic analysis. They named the new approach "Temporal-FPTN". The basic structure of the FPTN module was kept as it is in the classical form, however, temporal logic, referred to as Failure

Temporal Logic (FTL) is used instead of Boolean logic to specify equations to express the relationships between input and output failure modes to a FPTN module. As nothing is changed except the logic to represent equations, the new modelling framework is still suitable to be used in hierarchical structure, and now the outcome of the analysis would be temporal fault trees with Minimal Cut Sequences (MCSQs) instead of classical fault trees with minimal cut sets (see Figure 14).

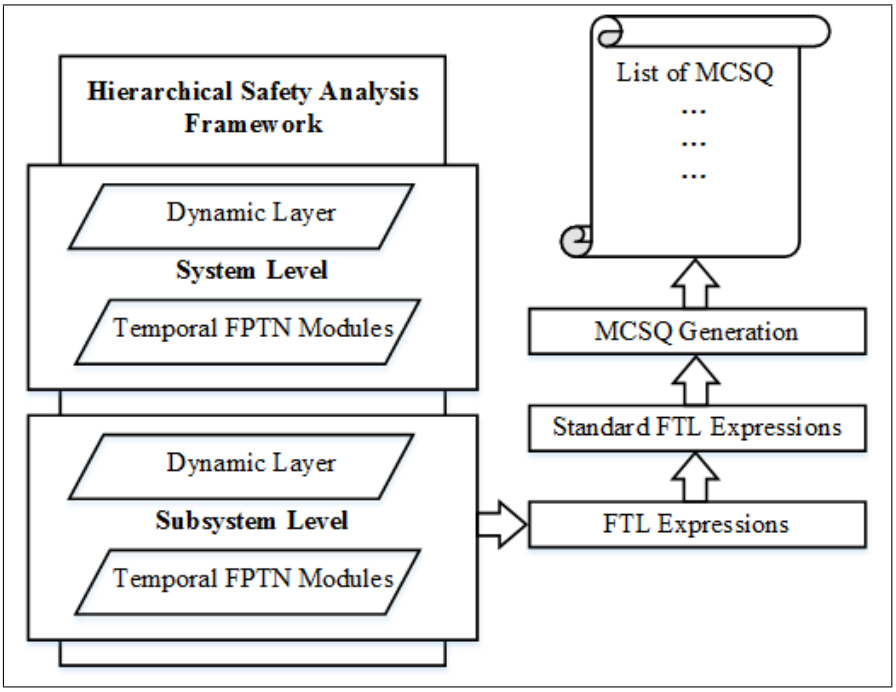


Figure 14: Safety Analysis Process using Temporal-FPTN

2.3.2 Failure Propagation and Transformation Calculus

Failure Propagation and Transformation Calculus (FPTC), closely related to FPTN, is a method for determining and modelling failure behaviour of both hardware and software components of systems (Wallace, 2005). Initially, it was designed to be used for the analysis of hard real-time software systems where the statistically schedulable code unit is considered as the primary unit of the architectural description of the system. Data and control flow behaviour of the system is defined by connecting these units using communications protocols like handshakes, buffers etc. One important thing to note is that FPTC treats communication protocols as similar to the system components, i.e., they have their own failure behaviour similar to system components and are modelled during the analysis. FPTC also assumes that all the threads and communications are known to the designers and are not created or destroyed dynamically during the system operation, i.e., the system architecture is considered as static.

Unlike the FPTN module, the FPTC represents the system architecture with a Real-Time Network (RTN) like notation as shown in Figure 15. A RTN graph consists of nodes and arcs representing software and software components and interactions between them.

FPTC generalizes the FPTN equations to determine how a component might respond to input failure modes to improve manageability and analysability. In addition to the classified failure modes of FPTN like value failures, timing failures, omission, and commission etc., it includes nominal behaviour (represented as *) as a new type which indicates lack of failure.

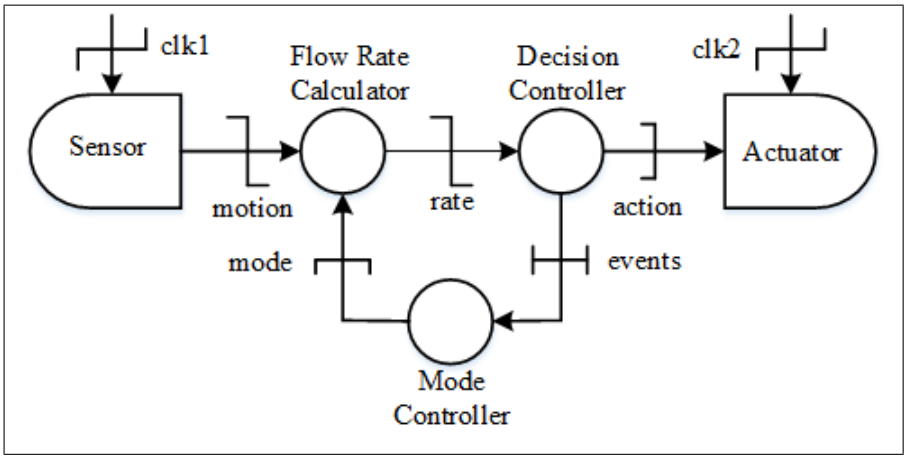


Figure 15: Example Real-Time Network Architecture

Similar to a FPTN module, a FPTC component can propagate failures (passing on input failure modes to output failure modes) or transform failures (mapping failure of one type to another) or a component itself can generate failures, or a component can detect and correct some failures (see Figure 16). The responses of a component to its input failure mode are expressed in a simple language based on patterns.

omission	→	omission (*propagating failure*)
late	→	value (*transforming failure*)
*	→	late (*generating failure*)
commission	→	* (*correcting failure*)

Figure 16: Behaviour of FPTC Component

FPTC can also handle multiple inputs/outputs by means of *tuples* of failures e.g.:

$$late \rightarrow (value, *, late)$$

The above expression represents a scenario where a component has a single input and three outputs. A timing failure, *late*, at the input will lead to a *value* failure at the first output, no detectable failure at second output, and a timing failure (*late*) at the third output. Equations for all output failures are specified considering all possible combinations of input failures which results in combinatorial explosion. In order to address this problem, Wallace (2005) has introduced the concept of using wildcards and variables to represent equations in a compact and intuitive format. The reason behind using wildcards and

variables is that many of the combinations of the inputs lead to a single output pattern. For example, following expressions show the use of wildcard and variable respectively.

$$(late,\ \_) \rightarrow (late,\ value)$$
$$(late,\ f) \rightarrow (f,\ late)$$

In the first expression, a wildcard (underscore symbol) is used and the expression means that a late failure at the first input and any failure at the second would lead to the same failure at the first and second output. A variable (f) is used in the second expression which means whatever failure present on the second input is propagated to the first output.

Unlike other safety analysis techniques, FPTC does not suggest the synthesis of fault trees from the FTPC models; rather it uses a fixed-point calculation technique. A distinct advantage of using FPTC over other safety analysis techniques is that it overcomes one significant limitation of other safety analyses techniques — handling cycles in architectures by using fixed-point calculations. An open source tool, Epsilon (Paige *et al.*, 2008), was introduced to implement FPTC and the tool relies on a generic meta-model to support transformations from AADL and SysML (Friedenthal *et al.*, 2014) models. Recently, a probabilistic variant of FPTC known as Failure Propagation and Transformation Analysis (FPTA) is proposed by Ge *et al.* (2009). This method links architectural models with probabilistic model checkers specified in PRISM (Kwiatkowska *et al.*, 2011) and allows FPTC to capture non-deterministic failure behaviour. FI^4FA (Gallina and Punnekkat, 2011) is the most recent extension of the FPTC which allows FPTC to consider more types of failure behaviour e.g., incompletion, inconsistency, interference and impermanence and also allows to analyse mitigation behaviours. FPTC uses the full architectural models of the system while specifying failure behaviour of components and therefore can adapt much more readily to changes. As a result, the failure behaviour model is synchronized with the system design. However, the outcome of the analysis relies on the analyst correctly identifying all the potential failure modes of interest and developing accurate models of the consequent failure transformation behaviours of individual components. As all possible combinations of failures are required to be considered, combinatorial explosion could be an issue while applying FPTC in safety analysis, even though some sort of mechanisms are proposed based on some assumptions to avoid combinatorial explosion.

2.3.3 Component Fault Trees

Conventional fault trees show logical connections between different faults and their causes. They show the hierarchy of faults rather than the architectural hierarchy of the system components, therefore it is difficult to map fault tree elements to the corresponding system components. Component Fault Trees is a modular version of the conventional Fault Tree approach that extends classical fault trees by using real components in the tree structure. In the CFT approach, smaller fault trees for each system component are defined and those component fault trees are arranged in a hierarchical structure according to the

architectural hierarchy of the system (Kaiser *et al.*, 2003). This enables closer synchronization of CFTs with the system model. Like classical fault trees, CFTs use Boolean gates such as AND, OR, and K-out-of-N gates. Moreover, CFTs use input output failure ports and internal failure events. Input and output ports are used to specify possible points of failure propagation and internal events are similar to that of the basic events of classical fault trees. CFTs differ from standard fault trees in that they allow multiple top events to be defined and represent repeating events only once. As a result, CFTs look more like a directed acyclic graph (Cause Effect Graph) than a tree like structure. However, standard fault tree algorithms could still be used to analyse CFTs both qualitatively and quantitatively as CFTs are still logical connections between faults and their causes. An example of a CFT is shown in Figure 17.

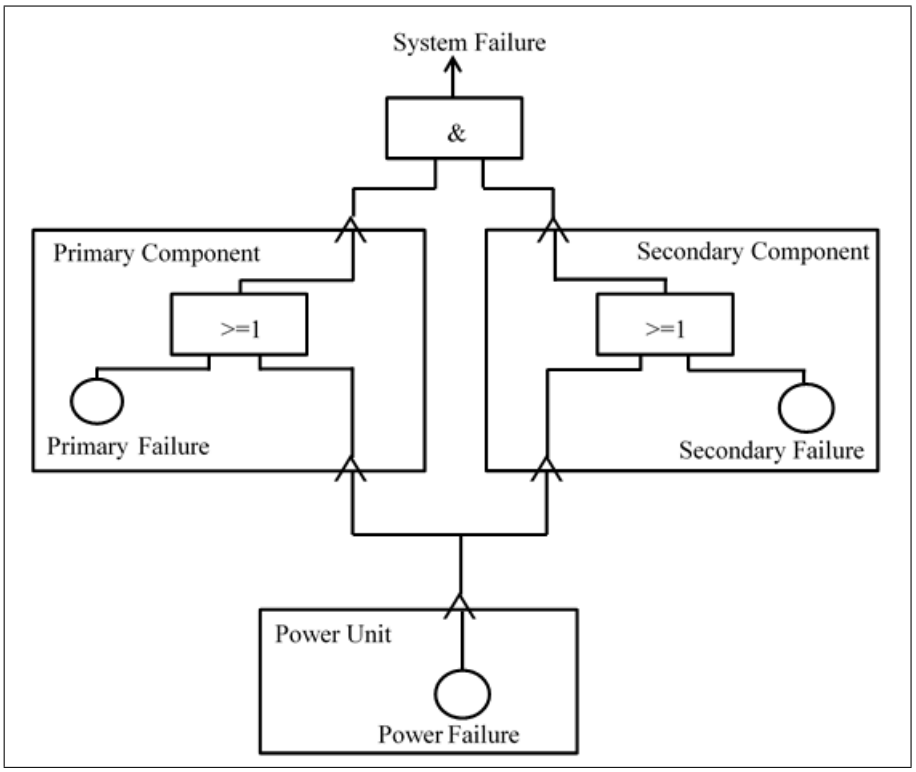


Figure 17: Example Component Fault Tree

The main advantage of using CFTs over classical FTs is its hierarchical decomposition of systems to manage the complexity of modern systems. Classical FTA cannot benefit from the decomposition facilities because while creating a standard fault tree, the analyst has to consider all levels of the system at once. In contrast, in CFT, smaller fault trees for each of the components are created and arranged in a hierarchical order according to the system architecture. As a result, different parts of the system can be developed and stored separately as part of the component definition in a library which facilitates a greater degree of reusability.

To be able to use CFTs in the architectural level of the systems, Grunske and Kaiser (2005) have presented an approach to construct a system level CFT hierarchically based on a system architecture where all the components of the architecture are annotated with

low-level CFTs. In order to analyse SaveCCM models (Carlson *et al.*, 2006; Åkerholm *et al.*, 2007), Grunske (2006) have extended the procedure of creating hierarchical CFTs. A windows-based tool, ESSaReL (ESSaRel, 2005) is available to perform minimal cut set analysis and probabilistic evaluation of CFTs. Recently another tool called ViSSaAn (Visual Support for Safety Analysis)(Yang *et al.*, 2011) has been developed based on a matrix view to allow improved visualisation of CFTs and efficiently representing information related to minimal cut sets of CFTs. Adler *et al.* (2011) have developed a meta-model to extract reusable CFTs from the functional architecture of systems specified in UML.

2.3.4 State Event Fault Trees

Even though FTA is a powerful technique widely used in reliability engineering, it has an inability to capture time or sequence dependent behaviour. This combinatorial technique is suitable for modelling static behaviour but is not suitable for modelling dynamic behaviour. To overcome this limitation, State-Event Fault Trees (SEFT) extend conventional FTA by adding capabilities for representing states (that last over a period of time) and events (sudden phenomena that cause state transition) to fault trees, thus overcoming the limitation of standard FTA by using state-based system models in the analysis (Grunske *et al.*, 2005). SEFTs can also be seen as an extension of CFTs with probabilistic finite state models. Therefore, elements/components in the system architecture are modelled with a set of states and probabilistic transitions between these states (see Figure 18). A state is graphically represented as a rounded rectangle and considered as a condition that lasts over a period of time, whereas an event is graphically represented as a solid bar and considered as an instantaneous phenomenon that can cause a state transition.

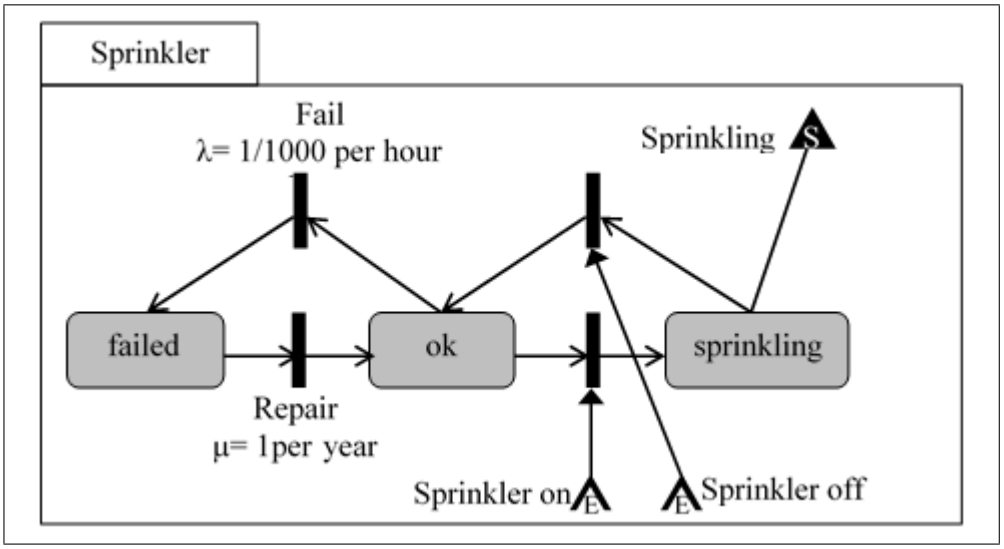


Figure 18: Example State Event Fault Tree

Grunske *et al.* (2005) have described a method to construct a SEFT from a system architecture where each of its elements has their SEFT defined. This method identifies the relationship between components based on name-matching of the state and event ports

as well as the data and control flow specified in the architecture.

As SEFTs make a distinction between causal and sequential transitions they therefore provide separate types of ports for them. Sequential transition applies to states which specify predecessor / successor relation between states; in contrast, causal transition applies to events which define a causal (trigger/guard) relationship between events. As events are explicitly represented in SEFTs, it is possible for one or more combinations of events to cause another event. To combine the events Boolean gates (e.g., AND and OR) of standard fault trees could be used. SEFTs can also include temporal information in the failure expression like assigning events with deterministic or probabilistic delays by using Delay Gates. History-AND, Priority-AND and Duration gates are employed to specify causal and sequential relations between events with respect to time. For example, the History-AND gate checks whether an event has occurred in the past, the Priority-AND gate can check in what order events have occurred, and the Duration gate can ensure that the system has been in a certain state for a specified amount of time.

Like CFTs, SEFTs can also be structured as a hierarchy of architectural components, where ports are used to specify the interactions between the components. SEFTs follow the same procedure as that of FTAs during modelling of system failure, i.e., the analysis starts with a system failure and works backwards to determine its root causes. As it uses state-based behaviour descriptions it can use pre-existing state-based models from the system design, which results in a greater degree of reusability compared to classical fault trees. However, it is no longer possible to analyse SEFTs using traditional fault tree algorithms because of the state-based representation they have used. Different types of techniques are now required to convert SEFTs into other representation like Petri Nets or Markov chains for quantitative evaluation of SEFTs. Steiner *et al.* (2012) have proposed a methodology to create and analyse SEFTs using the ESSaRel tool (ESSaRel, 2005). Consequently, the SEFT models are converted to Deterministic Stochastic Petri Nets (DSPNs) (Marsan and Chiola, 1987), then the analysis of the DSPN models can be performed using separate DSPN analyser like TimeNET (German and Mitzlaff, 1995). In the conversion process the whole system is required to be considered, i.e., all the components and subcomponents with their own state-based behaviour are to be considered, which would lead to a combinatorial state-space explosion. This problem can be remedied to some extent by using a combinatorial-FTA like algorithm for the static part of the system and using more efficient algorithms for the dynamic part of the system. However, the performance of the dual-analysis technique will depend on the type and complexity of the system being analysed.

2.3.5 HiP-HOPS

Hierarchically Performed Hazard Origin & Propagation Studies or HiP-HOPS (Papadopoulos, 2000; Papadopoulos and Mcdermid, 1999) is one of the more advanced and well supported compositional safety analysis techniques. It can provide similar functionality

to CFTs and FPTN but with more features and a greater degree of automation. It enhances the architectural design of a system with logical failure annotations to construct fault trees and FMEA automatically. It has some design optimisation capability that can help with decisions on component and subsystem selection among alternatives as well as decisions on the level and location of replicated components. It uses novel algorithms for top-down semi-automatic allocation of safety requirements in the form of Safety Integrity Levels which automates some of the processes for the ASIL allocation specified in the new automotive safety standard ISO26262 (ISO, 2011).

2.3.5.1 Overview of the Methodology

HiP-HOPS analysis consists of four phases:

1. System modelling and failure annotation
2. Fault Tree synthesis
3. Fault Tree analysis and FMEA synthesis
4. Optimisation

A generic overview of the HiP-HOPS safety analysis technique is shown in Figure 19. The first phase is performed manually and it consists of developing a model of the system and then annotating the system model with the failure behaviour to produce fault trees and perform the analysis. Modelling and annotation of the system can be done using modelling tools like Matlab Simulink. In the fault tree synthesis phase, the tool takes an annotated system model as input and use automated algorithms to create a series of fault trees which define failures along with their causes in the architecture.

The algorithm works deductively, i.e., it takes failure of system outputs and works backward through the system architecture to determine which components caused those failures. These failures are then joined together using the appropriate logical operators to construct fault trees, considering system failures as top events and lower level components failures (root causes) as basic events. In the third phase, fault trees generated in the previous phase are analysed and a FMEA is constructed combining the entire information stored in the different fault trees. At the same time both qualitative analysis and quantitative analysis are carried out on the fault trees, where the former provides the minimal cut sets and the latter provides the unavailability of the top event. If the system is required to be optimized then the optimisation is done iteratively in the last phase using genetic algorithms. A brief description of the each of the phases is given below. As model optimisation is out of scope of this thesis, the optimisation phase is not described.

2.3.5.2 Annotation Phase

HiP-HOPS can analyse any system that has identifiable components and some material, energy or dataflow transactions among the components. Before generating any fault trees

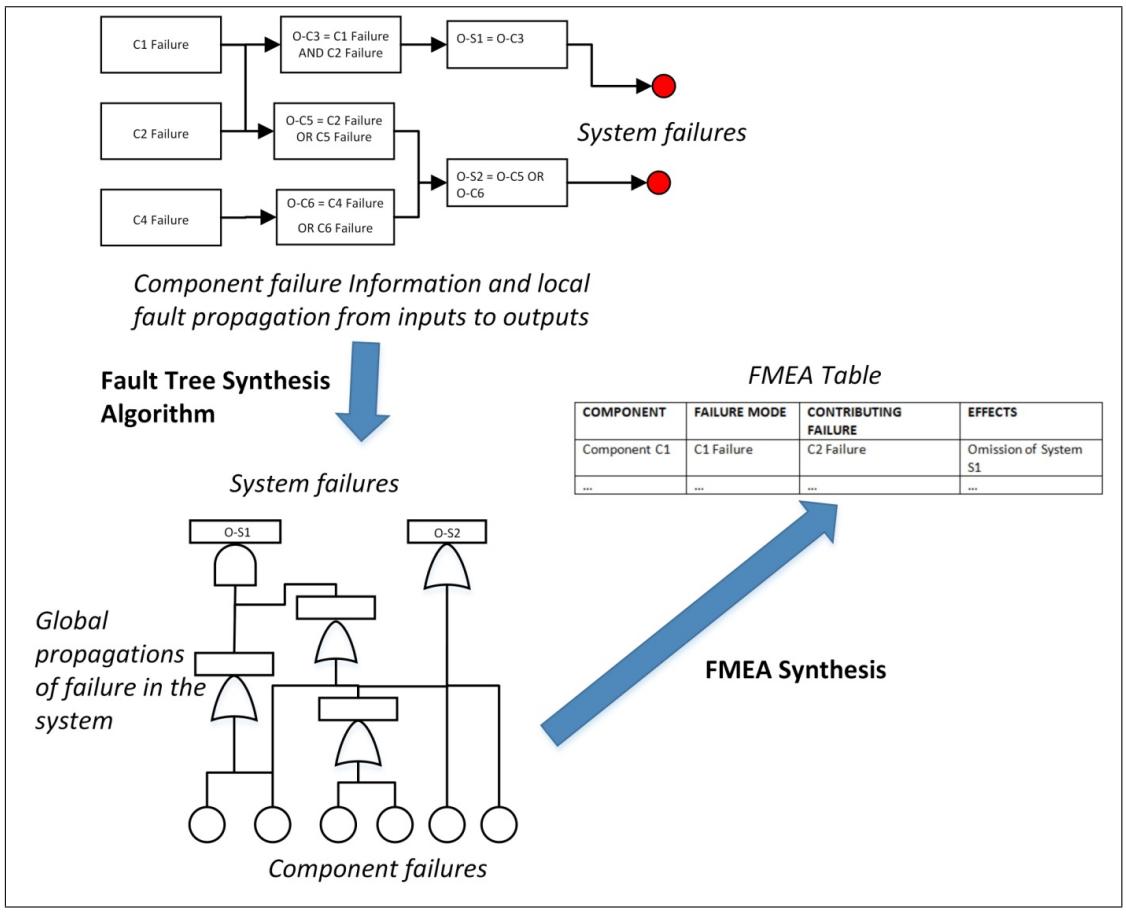


Figure 19: A generic overview of HiP-HOPS Technique

for system architecture, the tool needs to know how the different system components are interconnected and how they can fail. The architectural model of the system shows the interconnections between the components of system and the architecture can be arranged hierarchically, i.e., the system consists of different subsystems and subsystems have their own components (see Figure 20).

HiP-HOPS considers that an output failure of a component could be caused by an internal failure, an input failure, or combination of both. Therefore, local failure information for each of the components needs to be entered which describes what can go wrong with a component and how a component responds to a failure propagated from other parts of the system. This is achieved by annotating the model with a set of expressions showing how the deviations in the component outputs can be caused either by internal failure of that components or corresponding deviations in the component's input. Such deviations include omission of output, unexpected commission of output, incorrect output, or too late or early arrival of output. Other than the logical behaviour of the components, numerical data (failure rate and severity of the component) can also be entered for the components. Numerical data will then be used during quantitative analysis in the analysis phase. If any advanced analysis is planned then other data can be entered. When all the logical and numerical information regarding the failure behaviour of the components are specified, then the component can be stored in a library to allow a greater degree of

reusability. Failure modes and annotations of the failure modes of a simple model of a standby recovery system of Figure 20 are shown in Table 4 and 5 respectively.

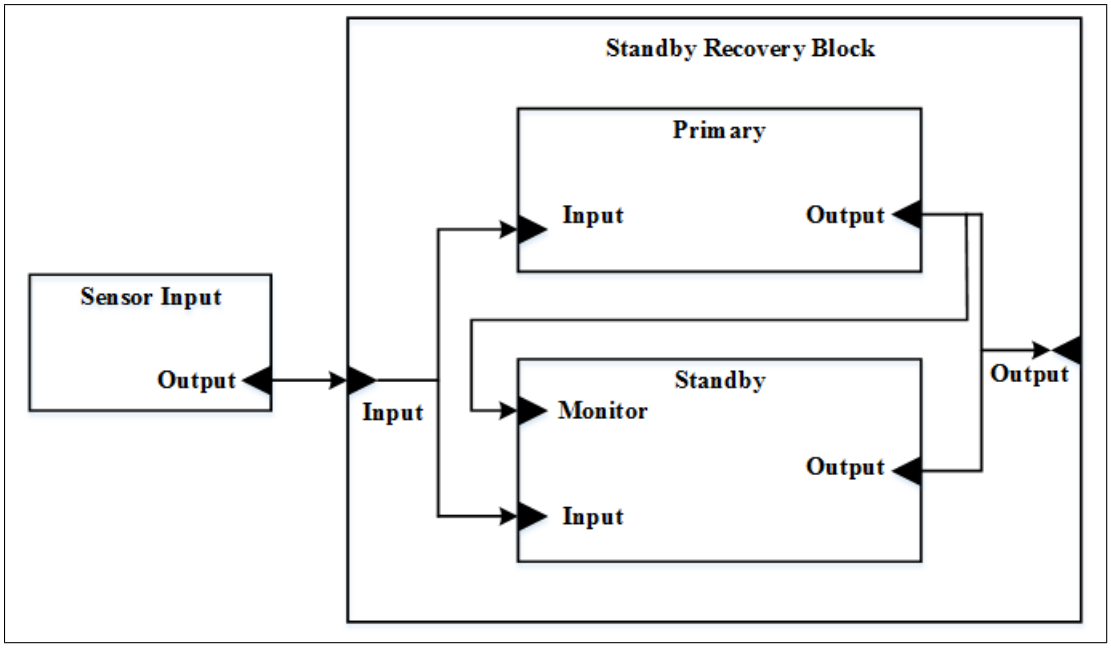


Figure 20: Example of Standby Recovery System

Table 4: Failure modes of Standby Recovery System

Component	Failure Modes	Probability
Standby Recovery Block	Electro Magnetic Interference	0.001
Primary	Internal Failure	0.03
Standby	Internal Failure	0.02
Sensor Input	Internal Failure	0.05

2.3.5.3 Synthesis Phase

As soon as local failure data for the components are entered, HiP-HOPS can then use the system model to synthesise fault trees. During the annotation process failure expressions are added manually to each component by the designers, which essentially represent CFTs of the components describing the propagation, generation, and transformation of failures between the inputs and outputs of the components. If a component is considered as a standalone module then the input and output deviations of that component are just the potential conditions of failure. But if we consider a component as a part of a system then its input deviations represent failure conditions which are caused by other components further upstream in the model, whereas the output deviations are the cause of failure further downstream in the model. That means output deviations of a particular class of one component at one end of a connection trigger input deviations to other components at the

Table 5: Failure Mode Expressions for Standby Recovery System

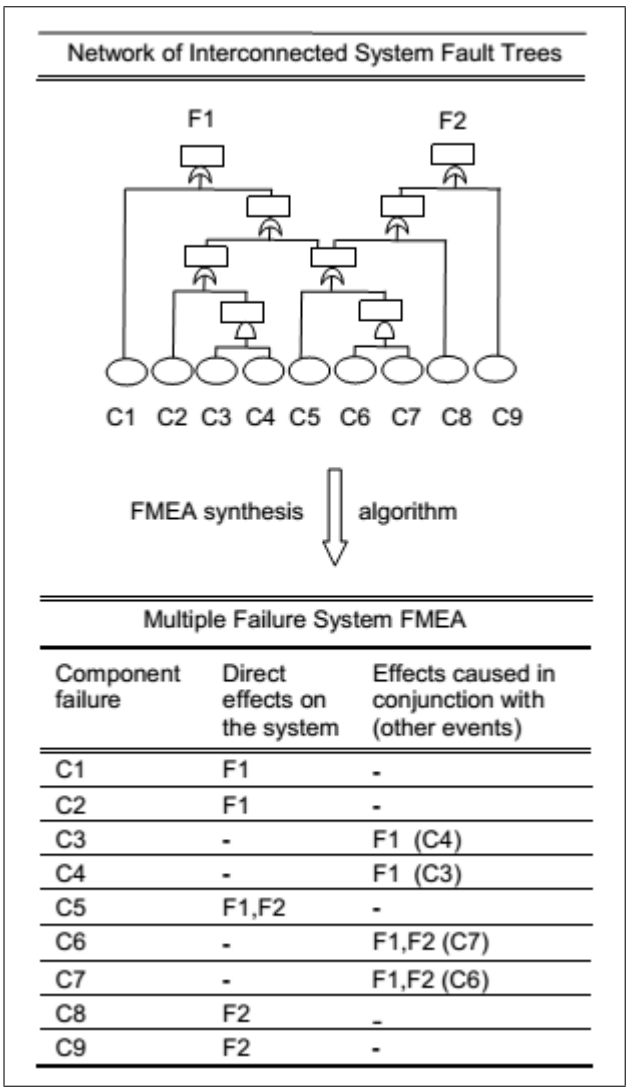
Component	Output Deviations	Failure expressions
Standby Recovery Block	Omission-Output	Omission-Primary.Output AND Omission-Standby.Output OR Electro Magnetic Interference
Primary	Omission-Output	Omission-Input OR Failure
Standby	Omission-Output	Omission-Monitor AND (Omission-Input OR Failure)
Sensor Input	Omission-Output	Failure

other end of the connection. In this way, failures are propagated globally in the model. Therefore the synthesis phase starts its operation with a top event (deviation of system output) and traverses the model by following the propagation of failures backwards from the top level of the system towards the basic component level. Fault trees are generated iteratively by replacing the input deviations of each component by the corresponding output deviations propagated by other components. This process continues till the basic events of the fault tree are found.

2.3.5.4 Analysis Phase

The synthesis process constructed a set of one or more interconnected fault trees, therefore in the analysis phase those synthesised fault trees are analysed both qualitatively and quantitatively, and eventually an FMEA is created from these results. Usually synthesised fault trees are large and complex; however, they can be minimised. Therefore, to reduce the size and complexity of fault trees, qualitative analysis is performed to obtain minimal cut sets. This is done by applying logical rules to reduce complex expressions, and there are many methods available to do so. HiP-HOPS uses a version of MICSUP (Pande *et al.*, 1975) as its primary minimal cut set generating algorithm. Once the minimal cut sets have been obtained, they are used for quantitative analysis to obtain the probability of the top event using basic events failure rate and equations (2) and (3).

In addition to the quantitative analysis, in the last step a further qualitative analysis is performed to generate an FMEA, which is a table concisely showing the results. The FMEA shows the direct connections between component failures and system failures, and hence it is possible to see both how a failure for a given component affects everything else in the system and also how likely that failure is. Although a classical FMEA only can



Multiple Failure System FMEA

Component failure	Direct effects on the system	Effects caused in conjunction with (other events)
C1	F1	-
C2	F1	-
C3	-	F1 (C4)
C4	-	F1 (C3)
C5	F1,F2	-
C6	-	F1,F2 (C7)
C7	-	F1,F2 (C6)
C8	F2	-
C9	F2	-

Figure 21: Conversion of FMEA from fault trees

show the direct effects of the single failure modes on the system, the FMEA generated by HiP-HOPS can also show the further effects of a failure mode, i.e. what are the possible effects of a failure mode if it occurs in conjunction with other failure modes? The concept of generating an FMEA from a fault tree using HiP-HOPS is shown in Figure 21.

In this example, C1-C9 are the basic events (component failures), and F1 and F2 are the system failures (top events). As seen in Figure 21, component failures C1 and C2 have direct effect on system failure F1 (and C8 and C9 have on F2). Components C3, C4, C6, and C7 have no direct effects on the system, i.e., individual failure of those components cannot cause system failure. However, in conjunction they have effects on the system, e.g., C6 and C7 both occurring in conjunction can cause F1 and F2 to occur. Common cause failure can also be identified from this FMEA, e.g., component C5 is causing both F1 and F2.

One important thing to note is that one common weakness of most of the model-based safety analysis techniques is their inability of generating FMEA. HiP-HOPS is one of the only approaches that provides the means to generate FMEA tables by analysing minimal

cut sets of the generated fault trees (Papadopoulos *et al.*, 2004a).

HiP-HOPS is supported by a mature tool (Papadopoulos, 2012) which can perform system safety analysis and architectural optimisation. The tool has been developed as an independent plug-in and has been interfaced to a range of tools, including Matlab Simulink and Simulation X. HiP-HOPS has been successfully applied by many companies like Volvo (Papadopoulos and Grante, 2005) or Daimler Chrysler (Papadopoulos *et al.*, 2001) to analyse many complex real-world systems and this technique is widely recognised as one of the state-of-the-art techniques in the reliability engineering area.

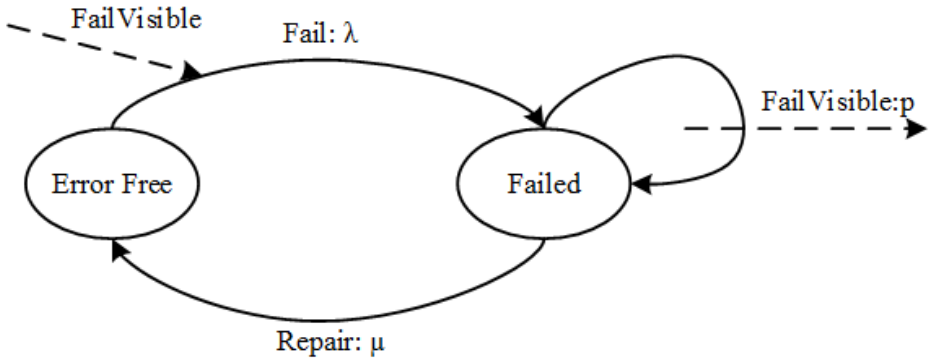
2.3.6 AADL

Architecture Analysis and Design Language (AADL) is a domain-specific language standardised by the International Society of Automotive Engineers (SAE) for the specification and analysis of hardware and software architectures of performance-critical real-time systems (SAE, 2012). The language has the capability to represent a system as an assembly of software components mapped onto an execution platform. The language provides a standard set of component abstractions, such as 'thread', 'thread group', 'process', 'data', and 'subprogram' for software components; 'processor', 'memory', 'device' and 'bus' for hardware components; and 'system' as a composite component (Feiler *et al.*, 2006a). AADL supports different types of interaction between components, for example events and dataflows, and the interactions between hardware and software components are defined through binding. Moreover, the language is extensible through customised "annexes", and the error model annex is one such extension of AADL using state machines. These are associated with an AADL component or connection to enable the specification of redundancy management and risk mitigation methods in architecture, and to enable qualitative and quantitative assessments of system properties such as safety, reliability, integrity, availability, and maintainability.

The dependability model consists of architecture modelling and error modelling. The architecture model, often referred to as the nominal architecture, describes system components with their connections and interactions. In contrast, the error modelling specifies the failure behaviour of AADL elements in terms of the propagation of *faults* as finite state machines consisting of finite number of states. Transitions between states are triggered by internal *error events* (either failure or repair events of components) or by *error propagation*. Error propagations are caused by error states of other components on the component in question. An AADL error model consists of a model type and at least one error model implementation. Error models can be associated with hardware components, software components, composite components and connections.

Figure 22 shows a graphical representation and textual description of an AADL error model of a component with propagations. As seen from Figure 22, the complete error model consists of two parts. The first part, the *error model* specifies the structure of the component and the second part, the *error model implementation* defines the behaviour

of the component. In this example the component has two states: *Error Free* (initial state) and *Failed*, two internal events *Fail* and *Repair*. The transition from the *Error Free* state to the *Failed* state can be caused by either internal failure event *Fail* or an incoming error propagation *FailedVisible* coming from the component's environment (other components). The component will generate an outgoing propagation with a fixed probability (p) while it is in the *Failed* state. The system component can go back to the *Error Free* state if the internal event *Repair* occurs. In this example, the internal events are associated with a *Poisson* probability distribution. The Occurrence properties for error events and error propagations specify the arrival rate or the occurrence probability respectively. Those properties can be declared in the error model type (considered as the default value) and in the error model implementation. Therefore, an occurrence value in the error model type can be replaced by a component specific value whose component is associated with the error model.




```
error model dependent
features
ErrorFree: initial error state;
Failed: error state;
Fail: error event{Occurrence => poisson λ};
Repair: error event{Occurrence => poisson μ};
FailedVisible: in out error propagation{Occurrence => fixed p};
end dependent;


error model implementation dependent.general
transitions
ErrorFree-[Fail]->Failed;
Failed-[Repair]->ErrorFree;
ErrorFree-[in FailedVisible]->Failed;
Failed-[out FailedVisible]->Failed;
end dependent.general;
```

Figure 22: AADL Error Model Definition for Component with Error Propagation

AADL components represent the elements of a system; therefore, systems may be represented as collections of components and hierarchies of components. As a result, the system error model can be considered as a composition of the error models of its components or subsystems. Each AADL error model can be stored in a library and can be reused for different AADL components. Interactions between components take place by propagating errors from the failing component to the dependent components. Propagation of errors between components is determined by their interdependencies and the AADL Error Model Annex has specified a set of **dependency rules** (Feiler and Rugina, 2007) to define interdependencies between components. For example, propagations may occur from a component to any outgoing connections and between all sub-components of the same process, which is conceptually similar to Papadopoulos's approach of integrating HiP-HOPS and Matlab Simulink models (Papadopoulos and Maruhn, 2001).

Figure 23 shows propagation between error model instances based on application component interactions and execution platform bindings. In the figure, the upper portion shows two application components are connected via a port connection where both components and port has their error model (state machines), and the error propagations occurs in the same direction of port flows. The lower portion of the figure shows two processors interconnected via a bus and errors are propagated between them due to the connectivity through the bus. Other than the error propagations between interconnected components, error propagation can occur between the hardware and the application components and connections, due to their binding to the execution platform. Propagations coming to a component or going out from a component can be filtered by using **Guard** properties. For details about the **Guard** properties readers are referred to (Feiler and Rugina, 2007).

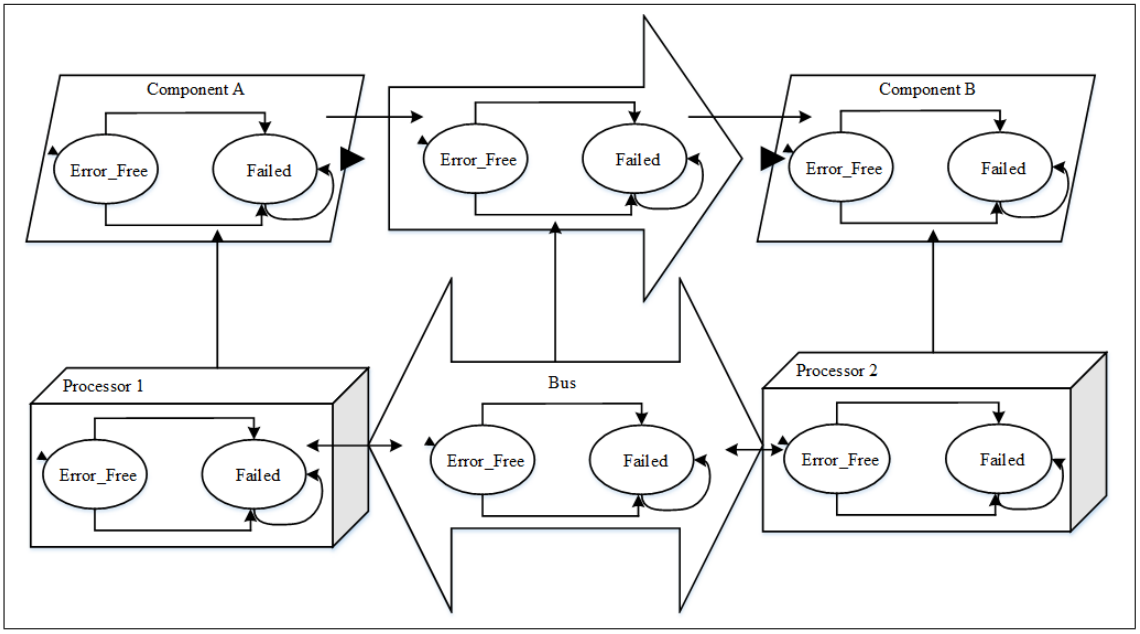


Figure 23: Error Propagations between Error Model Instances (Feiler and Rugina, 2007)

To analyse an AADL error model there are currently two approaches available. The first approach automatically translates an error model into a standard fault tree (Joshi

et al., 2007), and the second approach generates Generalized Stochastic Petri Nets (GSPNs) from error model specifications and uses existing GSPN tools for quantitative analysis (Rugina *et al.*, 2007). Recently, a number of researchers have integrated AADL and AltaRica Dataflow. For example, Mokos *et al.* (2008) have presented a method for transforming the AADL model to AltaRica model.

2.4 Behavioural Fault Simulation Approaches to MBSA

In behavioural fault simulation, system failure models equivalent to an FMEA are produced by injecting faults into executable formal specifications of a system, thereby establishing the system level effects of faults. In most cases, the results are then used in a model-checker to verify the system safety properties. Techniques that follow this approach include FSAP-NuSMV, Altarica, DCCA, and SAML. These different techniques will be discussed in the following subsections.

2.4.1 FSAP/NuSMV

The Formal Safety Analysis Platform FSAP/NuSMV-SA (Bozzano and Villafiorita, 2003, 2007) consists of a set of tools including a graphical user interface tool, FSAP, and an extension of model checking engine NuSMV. The aim of this platform is to support formal analysis and safety assessment of complex systems. This platform allows the user to inject particular failure modes into the system (either nominal mode or degraded mode of the system) and then observe the effects of that failure on the system behaviour. The primary goal of the FSAP/NuSMV-SA toolset is to verify if a system model is satisfying its safety requirements; however, it is capable of performing different types of safety analyses, e.g., automatic fault tree generation.

The components of the FSAP/NuSMV-SA platform are shown in Figure 24. As seen in the figure, it has different modules to perform different tasks. The central module of the platform is the *SAT Manager* which can control the other modules of the platform. It stores all the information related to safety assessment and verification which includes references to system model, extended system model, failure modes, safety requirements and analyses. System models are described as finite state machines with the NuSMV language in the *Model Capturing* module as plain text. This model can be a formal safety model or a functional system model and the user has the flexibility to use their preferred text editor to design or edit the system model. An example of a NuSMV model of two-bit adder is shown in Figure 25 (originally used in (Bozzano and Villafiorita, 2007)). The *Failure Mode Editor* and the *Fault Injector* modules allow the user to inject failure modes in the system model to create an extended system model (see Figure 26). The expressions of the failure modes can be stored in a library to provide greater degree of reusability. The system model is then augmented with safety requirements in the *Safety Requirement Editor*. Temporal logic is used to express the safety requirements and can also be stored in

a library to facilitate future reuse. The *Analysis Task Manager* defines the analysis tasks that are required to be performed, i.e., specification of the analyses. The next step is to assess the annotated system model against its functional safety requirements. This task is done based on model checking technique in the *NuSMV-SA Model Checker* module. This module also generates counter examples and safety analysis results by means of fault trees etc. The *Result Extraction and Displayers* module process all the results generated by the platform and present to the user. The fault trees can be viewed in the displayer that is embedded in the platform or using commercial tools, and counter examples can be viewed in a textual or graphical or tabular fashion etc.

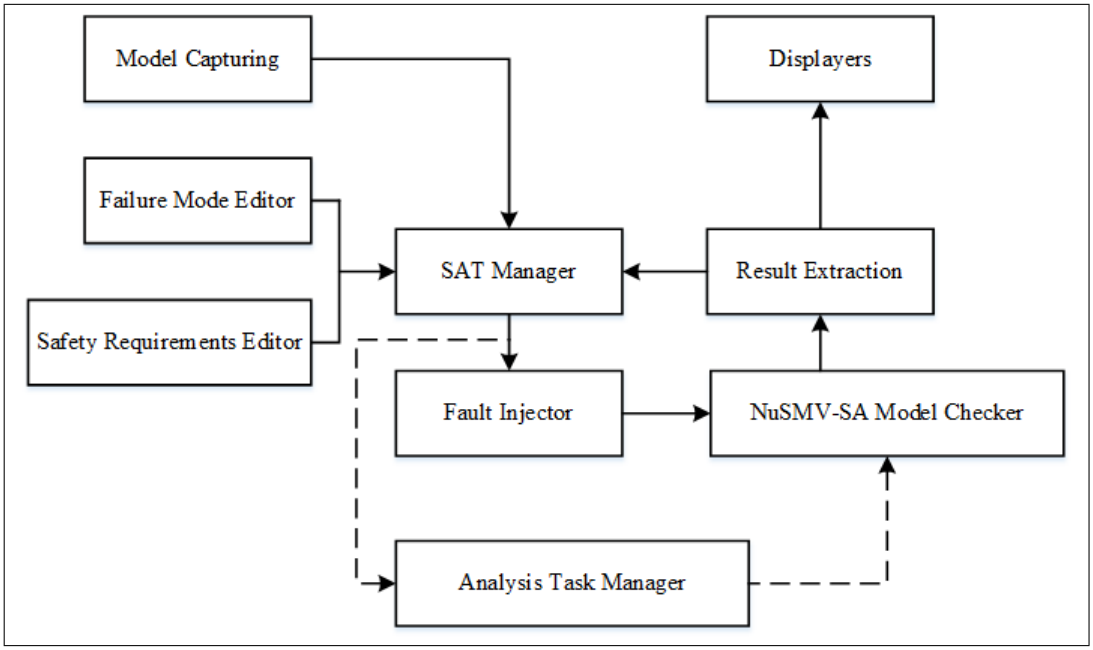


Figure 24: Components of FSAP/NuSMV-SA Platform

In the FSAP/NuSMV-SA, fault trees are generated step by step. At first a reachability analysis is performed by starting with the initial states of the system model and traversing (by iteratively creating a set of successor states) the system state chart until a state is reached in which the top event of the fault tree has occurred. The outcome of this process is a set of all states where failures of components occur. The minimal cut sets are obtained from these set of states by extracting only the failure variables from these states and writing the expressions for those failure modes based on the information obtained from the state machine. As standard FTs are combinatorial this process ignores dynamic information stored in the system model. However, to preserve the dynamic information NuSMV-SA can add traces with each cut set describing how the failure can occur. Another way of doing the dynamic analysis is to perform the ordering analysis on minimal cut sets (Walker *et al.*, 2008).

Although the FSAP/NuSMV-SA platform provides multiple functionality, it does also have some limitations, especially in handling fault trees. Fault trees generated by this toolset show the relation between top events and basic events; however, how the faults are propagated through the different level of the system model is not shown which could

```
MODULE bit(input)
VAR
     output : {0,1};
ASSIGN
output := input;
MODULE adder(bit1, bit2)
VAR
     output : {0,1};
ASSIGN
     output := (bit1 + bit2) mod 2;

MODULE main
VAR
     random1   : {0,1};
     random2   : {0,1};
     bit1    : bit(random1);
     bit2    : bit(random2);
     adder    : adder(bit1.output, bit2.output);
```

Figure 25: A NuSMV model for two-bit adder (Bozzano and Villafiorita, 2007)

```
VAR    output_nominal        : {0,1};
      output_FailureMode          : {no_failure, inverted};
ASSIGN output_nominal  := input;
DEFINE output_inverted := ! output_nominal;
DEFINE output := case
    output_FailureMode = no_failure : output_nominal;
    output_FailureMode = inverted : output_inverted;
 esac;
ASSIGN next(output_FailureMode) := case
      output_FailureMode = no_failure : {no_failure, inverted};
 output_FailureMode = inverted : inverted;
      esac;
```

Figure 26: Injecting fault into the bit module

make the fault trees less visibly understandable for complex systems. It provides the means to perform qualitative FTA, but it does not have the ability to perform probabilistic evaluation of FTs. Like other state machine based approaches, this platform also suffers from state space explosion while modelling large or complex systems.

2.4.2 AltaRica

AltaRica is a high level description language (Arnold *et al.*, 2000; Point and Rauzy, 1999) based on finite state machines designed to model both nominal and failure behaviour of complex systems. It can represent systems as hierarchies of components and subcomponents and model both state and event like State-Event fault trees. Once a system has been modelled in AltaRica, it can be analysed by external tools and methods, e.g., the generation of fault trees, Petri nets, model-checking etc. (Bieber *et al.*, 2002).

In AltaRica, components are represented as nodes, and each node possesses a number of state and variables (either state variables or flow variables). The number of state and flow variable are discrete. The values of the state variables are local to the node they are in, and value of a state variable changes when an event occurs, i.e., events are triggering state transitions, thus changing the values of state variables. Flow variables are visible both locally and globally, and are therefore used to provide interfaces to the nodes.

Each basic component is described by an interfaces transition system, containing the description of the possible events, possible observations, possible configurations, mappings of what observations are linked to which configurations, and what transitions are possible for each configuration. A classic example of a component (electrical switch) in AltaRica is shown in Figure 27.

```
node Swith
 flow
   f1, f2 : bool;
 event
   open, close;
 state
   IsClosed : bool;
 trans
   not IsClosed |- close -> IsClosed := true;
   IsClosed |- open -> IsClosed := false;
 assert
   IsClosed => (f1 = f2)
edon
```

Figure 27: Node Example in AltaRica: Switch (Point and Rauzy, 1999)

The behaviour of a component (node) is defined through assertions and transitions. Assertions specify restrictions over the values of flow and state variables whereas transitions determine causal relations between state variables and events, consisting of a single trigger (event) and a guard that constraints the transition; guards are basically some assertions over flow and state variables. In the above example, the switch has a Boolean state variable *IsClosed*; transitions between the values (*true* or *false*) of this variable are triggered by the events *open* and *close*. The component has two flow variables: $f1$ and $f2$, the assertions over those two variables species that the power on both terminal of the switch will be same when the switch is closed.

After defining the nodes, they can be organized hierarchically to reflect system decomposition and architecture. The top-level node represents the system itself, and it consists of other lower-level nodes. Nodes can communicate either through interfaces or through event dependencies. The first process is done by specifying assertions over interfaces and the second one is done by defining a set of *broadcast synchronisation vectors*. These broadcast synchronisation vectors allow events in one node or component to be synchronised with those in another. These vectors can contain question marks to indicate that an event is not obligatory (e.g., a bulb cannot turn off in response to a power cut if it is already off). Additional constraints can be applied to the vectors to indicate that certain combinations or numbers of events must occur, particularly in the case of these 'optional' events, e.g., that at least one of a number of optional events must occur, or that k-out-of-n must occur etc.

Two main variants of AltaRica have been designed so far. The primary difference between the variants is how the variables are updated after firing of transitions. In the first version (Arnold *et al.*, 2000; Point and Rauzy, 1999), variables are updated by solving constraints, thus consuming too many resources. Therefore, this approach is not scalable for industrial application although it is very powerful. To make AltaRica capable of assessing industrial scale applications, a second version, AltaRica Data-Flow (Boiteau *et al.*, 2006; Rauzy, 2002) is introduced where variables are updated by propagating values in a fixed order, and the order is determined at compile time. This approach takes fewer resources than the first approach, however, it cannot naturally model bidirectional flows through a network, cannot capture located synchronisation, and faces difficulties in modelling looped systems. Currently, the third version of AltaRica, AltaRica 3.0 (Batteux *et al.*, 2013) is under specification. It improves the expressive power of the second version without reducing the efficiency of assessment algorithms. The main improvement is that it defines the system model as a new underlying mathematical model — Guarded Transition Systems (GTS) (Batteux *et al.*, 2013) — which allows analysts to model systems consisting of loops, and can easily model bidirectional flows. AltaRica 3.0 provides a set of assessment tools, e.g., Fault Tree generator, Markov chain generator, stochastic and stepwise simulator. However, it still lacks capability for capturing dynamic or temporal ordering of events in the generated Fault Trees.

2.4.3 DCCA

Deductive Cause Consequence Analysis or DCCA (Güdemann *et al.*, 2007; Ortmeier *et al.*, 2005) is a formal method for safety analysis. It determines whether a given component fault is the cause of a system failure or not by using mathematical methods. It is a formal generalization of FMEA and FTA, more formal than FTA and more expressive than FMEA. The goal of DCCA is:
"Given an unwanted, hazardous situation H and a set of component failure modes Γ*. Determine, which combinations of failures modes may (1) potentially cause a hazard and (2)*

are minimal in the sense, such that no proper subset of these failure modes can cause the hazard." (Güdemann *et al.*, 2007).

DCCA represents the system model as finite state automata with temporal semantics using Computational Tree Logic (CTL). It assumes that all the basic component failure modes are available, and then defines a set of temporal properties that indicate whether a certain combination of component failure modes can lead to the system failure. This property is known as the *criticality* of a set of failure modes which are analogous to cut sets of classical fault trees. Similar to FTA, DCCA aims at determining the minimal critical sets of failure modes which are necessary and sufficient to cause the top event (system failure) and therefore FTA may be considered as a special case of DCCA.

DCCA also faces the state explosion problem because to determine minimal critical sets it has to consider all possible combinations of component failure modes. This problem can be alleviated to some extent by using results from other informal safety analysis techniques like FTA which are believed to be generating smaller but good initial guesses for solutions. However, by doing this DCCA inherits the shortcomings of FTA, i.e., it is unable to capture dynamic behaviour where order of events is equally important as combination of failures. Deductive Failure Order Analysis (Güdemann *et al.*, 2008), a recent extension of DCCA enables it to deduce temporal ordering information of critical sets from DCCA. In this extension, Pandora style temporal gates like PAND and SAND are used to capture temporal behaviour. Temporal logic laws are also provided to make the temporal ordering deduction process automated.

2.4.4 SAML

Safety Analysis Modelling Language (SAML) (Güdemann and Ortmeier, 2010) is a tool-independent modelling framework that can construct system models with both deterministic and probabilistic behaviour. It utilises finite state automata with parallel synchronous execution capability with discrete time steps to describe a system model consisting of hardware and/or software components, environmental conditions etc. In the state automata, transitions can be defined both as probabilistic and non-deterministic. From a single SAML model both qualitative and quantitative analysis can be performed. An example of a SAML model is shown in Figure 28. A SAML model consists of at least one module description and declarations of zero or more constants (either of type double, int, or bool) and formulas. The example SAML model has two modules (A and B), four constants, and one formula. After declaring all the optional constants the module declaration starts with the keyword *'module'* followed by the module name and the declaration end with the keyword *'endmodule'*. As the modules are declared as state automata, every module has at least one state variable and at least one rule to update the state variable. Every state variable is represented as a range of integer values and initialised with a value. Every update rule defines either at least a probabilistic assignment or at least one non-deterministic choice of assignments, and they are conditioned on a Boolean activation

```
constant double P_A := 0.1;
constant double P_B1 := 0.2;
constant double P_B2 := 0.3;
constant double P_B3 := 0.5;
formula CASE_3 := V_A=0& ! (V_B1=0&V_B2=0|V_B1=1&V_B2=1)

module A
V_A:[0..2] init 0;
V_A=0&V_B1=0&V_B2=0 ->
choice (P_A:(V_A'=0)+(1-P_A):(V_A'=1));
V_A=0&V_B1=1&V_B2=1 -> choice (1:(V_A'=2));
CASE_3 -> choice (1:(V_A'=1));
V_A=1 -> choice (1:(V_A'=1));
V_A=2 -> choice (1:(V_A'=2));
endmodule

module B
V_B1:[0..1] init 0;
V_B2:[0..1] init 0;
true -> choice (P_B1:(V_B1'=0)&(V_B2'=0) +
P_B2:(V_B1'=1)&(V_B2'=0) +
P_B3:(V_B1'=1)&(V_B2'=1)) +
choice (1:(V_B1'=1)&(V_VB2'=1));
endmodule
```

Figure 28: Example SAML model (Güdemann and Ortmeier, 2010)

condition.

SAML models can be transformed automatically to the input format of other model-based safety analysis techniques without changing the architecture of the systems. Therefore SAML can work as an intermediate language for MBSA techniques, i.e., if models designed in any other higher-level language can be converted to SAML models (extended system models) then the resultant models can be transformed to input format of other targeted analysis tools, and, thereby analysed with all targeted tools.

Güdemann and Ortmeier (2011) have shown the ways of transforming SAML models into the input language of probabilistic model checker PRISM (Kwiatkowska *et al.*, 2011). In the same work, the above researchers also have shown the ways of transforming SAML modules to NuSMV modules although the former supports both non-deterministic and probabilistic update rules whereas the latter supports only non-deterministic update rules.

The primary advantage of SAML is its formal qualitative and quantitative semantics that allows different analyses to be performed in the same model. To perform quantitative analysis, many analysis techniques rely on the result of the previously performed qualita-

tive analysis but SAML can perform quantitative analysis independently (no reliance on qualitative analysis). However, SAML also suffers from the state explosion problem like other state-transition based formalism.

2.5 Other Related Techniques in Reliability Engineering

2.5.1 Markov Models

Markov Model analysis is another state-transition based graphical modelling technique used to capture the dynamic behaviour of systems. A Markov model of a system consists of a list of all possible states of the system, the transition among states and numerical values defining rate of each transition. In reliability engineering, the rates associated with each transition are usually the failure or repair rate of components. States are graphically represented as circles and directed edges are used to represent transition. Each state in the Markov model encompasses all the system information with reference to component failures, sequence of failures, and information on spare allocations (Manian *et al.*, 1998) and the probability of the system being in state i at time t is denoted by $P_i(t)$. A very simple Markov model of a system with two states: *Operational* and *Failed* is shown in Figure 29.

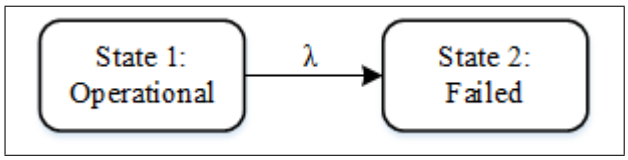


Figure 29: Simple example of Markov Model

In the figure, the rate of transition from state 1 to state 2 is λ, i.e., failure rate of the system is λ. After forming the graphical model, it is necessary to write ordinary differential equations (ODEs) for each state based on incoming and outgoing transitions which represent overall probability of the system being in a particular state. The solution of a Markov model is obtained by solving the differential equations of all states (Vesely *et al.*, 2002). For example, let us consider a generic parallel system with two independent, non-repairable components A and B and their failure rate is λ_A and λ_B respectively. The block diagram of the system and Markov model of the system are shown in Figure 30.

In state 1, the system is fully functional where both the components are available. In state 2, the system is operating with component B (with A failed) and in an opposite way in state 3 the system is operating with component A (with B failed). The final state F represents a state where both the component failed.

If we define the probability of the system being in states 1, 2, 3 and F as P_1, P_2, P_3 and

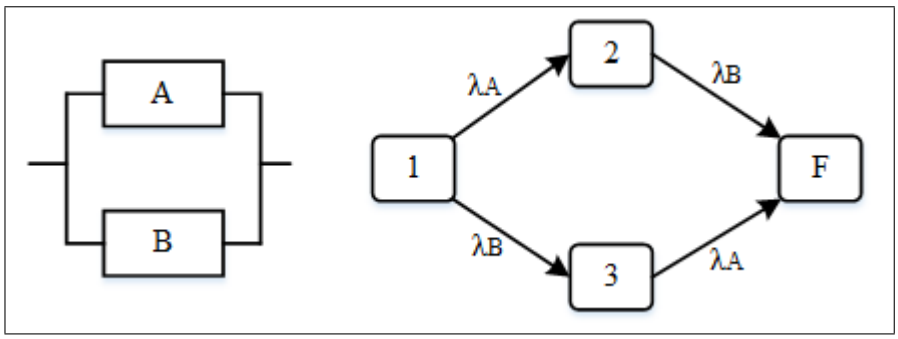


Figure 30: Non-repairable parallel system and Markov model of the system (Mahmud, 2012)

P_F then differential equations for the Markov model can be written as (Mahmud, 2012):

$$\begin{aligned}
\frac{d}{dt} P_1(t) &= -(\lambda_A + \lambda_B)\, P_1(t) \\
\frac{d}{dt} P_2(t) &= \lambda_A\, P_1(t) - \lambda_B\, P_2(t) \\
\frac{d}{dt} P_3(t) &= \lambda_B\, P_1(t) - \lambda_A\, P_3(t) \\
\frac{d}{dt} P_F(t) &= \lambda_A\, P_3(t) + \lambda_B\, P_2(t)
\end{aligned} \tag{12}$$

In the above equations for different states, on the right hand side, terms with negative sign represent the rate at which the system is leaving that particular state and terms with a positive sign imply the rate at which the system is entering that particular state. The solution of the Markov model is obtained by solving the above equations and can be written as:

$$\begin{aligned}
P_1(t) &= e^{-(\lambda_A+\lambda_B)t} \\
P_2(t) &= e^{-\lambda_B t} - e^{-(\lambda_A+\lambda_B)t} \\
P_3(t) &= e^{-\lambda_A t} - e^{-(\lambda_A+\lambda_B)t} \\
P_F(t) &= e^{-(\lambda_A+\lambda_B)t} - e^{-\lambda_A t} - e^{-\lambda_B\, t} + 1
\end{aligned} \tag{13}$$

Coverage modelling, i.e., modelling systems considering covered and uncovered failures of components as mutually exclusive events, which is difficult to integrate into FTA, can be easily incorporated into Markov models. After introducing the concept of coverage modelling, Markov models gradually replaced FTA as a method for reliability analysis of safety-critical fault tolerant systems (Dugan *et al.*, 2000). The Dynamic Fault Tree (DFT) approach is introduced by extending conventional FT with dynamic gates to provide a way to analyse FTs using Markov models (Vesely *et al.*, 2002; Dugan *et al.*, 1993). Boudali *et al.* (2007, 2010) have proposed input/output interactive Markov Chains to solve some issues in application of DFTs, like a lack of formal syntax and semantics, and limitations in modular analysis. Guo and Yang (2008) have proposed a technique to automatically generate Markov chains based on voting, self-diagnostic and failure mode information of

a system without using any intermediate model e.g., DFT. Due to the reputation of Markov Chains for quantitative safety analysis, different researchers like Aneziris and Papazoglou (2004); Bukowski and Goble (1995); Jianzhong and Julian (2011); Platis *et al.* (1998); Tao (2009); Wan and Wu (2009); have used Markov analysis method for dependability analysis of different systems.

Although Markov analysis is a popular graphical model for reliability analysis, the major shortcoming of Markov modelling is that for large and complex systems, the Markov model becomes exceptionally large and complicated and therefore difficult to construct and analyse. One more thing to note about the Markov chain model is that it defines the state transitions with exponentially distributed rates, i.e., it can model the system behaviour if the system components have exponential failure rate. As a result, if the system components have failure rates with any other distributions (e.g., Weibull distribution) then Markov chains cannot be used to analyse such systems.

2.5.2 Bayesian Networks

Bayesian Networks (BNs) are directed acyclic graphs that represent a set of random variables and their conditional dependencies (Pearl, 1988). Therefore a BN is made up of:

1. A Directed Acyclic Graph
2. Conditional Probability Table (CPT) for each node of the graph

A simple example of Bayesian Network and set of tables for each node is shown in Figure 31 and Table 6. The nodes of the graph represent the random variables (events) and arcs represent dependencies or cause-effect relations among nodes.

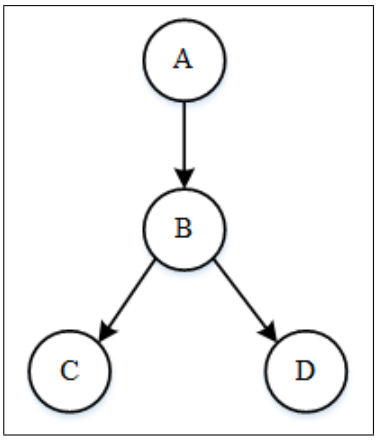


Figure 31: Simple example of Bayesian Network

In a Bayesian Network, a node X is said to be the parent of another node Y if there is an arrow from node X to node Y e.g., A is a parent of B and B is child of A in Figure 31. Parent nodes have a direct effect on its child nodes and each node X_i has a conditional probability distribution $Pr\{X_i|Parents(X_i)\}$ that quantifies the effect of the parents on the node. If a child node represent a Boolean variable and its parents are m Boolean variables

Table 6: Set of conditional probability table of BN in Figure 31

A	P(A)
true	0.4
false	0.6

A	B	P(B|A)
true	true	0.3
true	false	0.7
false	true	0.99
false	false	0.01

B	C	P(C|B)
true	true	0.1
true	false	0.9
false	true	0.6
false	false	0.4

B	D	P(D|B)
true	true	0.95
true	false	0.05
false	true	0.98
false	false	0.02

then the probability function could be represented by a table of 2^m entries, one entry for each of the 2^m possible combinations of its parents being true or false. A node without a parent is known as root node, and a node without child is called leaf node e.g., A is a root node and C and D are leaf nodes in Figure 31.

A set of conditional independence statements are the main considerations when making BN models and conditional independence information can be obtained from a BN model by employing the rules of *d-separation* (Pearl, 1988). Two random variables X and Y are conditionally independent given a third random variable Z if and only if they are independent in their conditional probability distribution given Z i.e., X and Y are conditionally independent given Z if and only if, given any value of Z, the probability distribution of X is the same for all values of Y and the probability distribution of Y is the same for all values of X.

Using the conditional independence assumptions of BNs, the joint probability distribution of a set of random variables $\{X_1, X_2, X_3, \cdots, X_{n-1}, X_n\}$ can be determined using a chain rule as explained in (Mi *et al.*, 2012; Chen *et al.*, 2010; Pearl, 1988):

$$Pr\{X_1, X_2, X_3, \cdots, X_{n-1}, X_n\} = \prod_{i=1}^{n} Pr\{X_i | \ Parent(X_i)\} \tag{14}$$

If A and B are two random events and evidence is found that event B has occurred then using the Bayes theorem, the posterior probability of event A on condition that B has happened can be obtained as:

$$Pr\{A|B\} = \frac{Pr\{B|A\} \ Pr\{A\}}{Pr\{B\}} \tag{15}$$

where $Pr\{A\}$ and $Pr\{B\}$ are the prior probability of event A and B respectively.

As BNs provide a robust probabilistic method of reasoning under uncertainty and they are capable of combining different sources of information to provide a global safety assessment, they have therefore received much attention in last decade in the area of dependability analysis (Bobbio *et al.*, 2001; Langseth and Portinale, 2007). Weber *et al.* (2012) have reported a continuous growth of the number of publications showing scientific and industrial interest for BNs. According to their observation, 61% of publications are about the application of BNs in dependability analysis, 26% are for risk analysis and 13% are in maintenance area as shown in Figure 32.

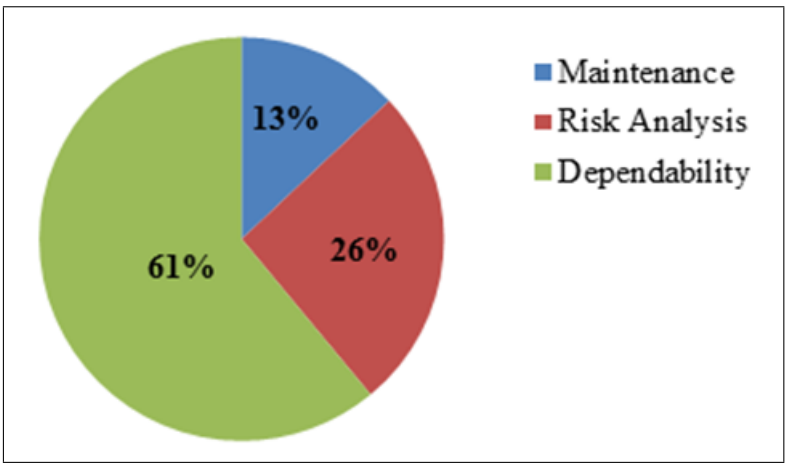


Figure 32: Distribution of use of BNs in reliability Engineering (Weber *et al.*, 2012)

Usually Bayesian Networks are used in two different ways to perform dependability analysis of systems. In the first way, systems are modelled using Bayesian Networks and then the analysis is performed on that model. Researchers like Doguc and Ramirez-Marquez (2009); Huang *et al.* (2008); Neil and Marquez (2012); Neil *et al.* (2008); Torres-Toledan and Sucar (1998) etc. have shown the ways of modelling systems directly in Bayesian Networks, and subsequent analysis on that system model. The second approach of using BNs in dependability analysis is to transform other representations like FTs, DFTs into BNs. For example, Bobbio *et al.* (2001) has introduced a way of translating FTs to BNs. They represented the basic events of the FTs as the root nodes of the BNs and the intermediate events (Boolean gates) as the intermediate nodes of the BNs. As the Boolean gates of the FTs represent deterministic causal relationships between the input and output events, the non-root nodes are actually deterministic nodes and all the entries in the CPT are either 0 or 1 (see Figure 33). And the prior probabilities of the root nodes are same as the probabilities assigned to the basic events (leaf nodes) in the FTs.

Using similar concepts, DFTs can also be translated into BNs to evaluate the dependability of the systems. For example, methods of transforming dynamic fault trees (DFT) to dynamic Bayesian network (DBN) are presented by Montani *et al.* (2006); and Boudali and Dugan (2005). To allow reliability analysis based on automatic translation of DFT to DBN, a tool called RADYBAN (Reliability Analysis with DYnamic BAyesian Networks) was developed by Montani *et al.* (2008).

Hybrid Bayesian Network (HBN) and Continuous-time Bayesian Networks (CTBN) were proposed by Neil *et al.* (2008) and Boudali and Dugan (2006) respectively to introduce continuous variables along with discrete variables in BNs. Neil and Marquez (2012) have used HBN to perform reliability analysis of systems with repairable components and the way of measuring importance of system components to determine their sensitivity using BNs is proposed by Zhong and Li (2013).

Not all the system failures are caused only by mechanical failure of system components or software failure: human error is an important constituent of system failure.

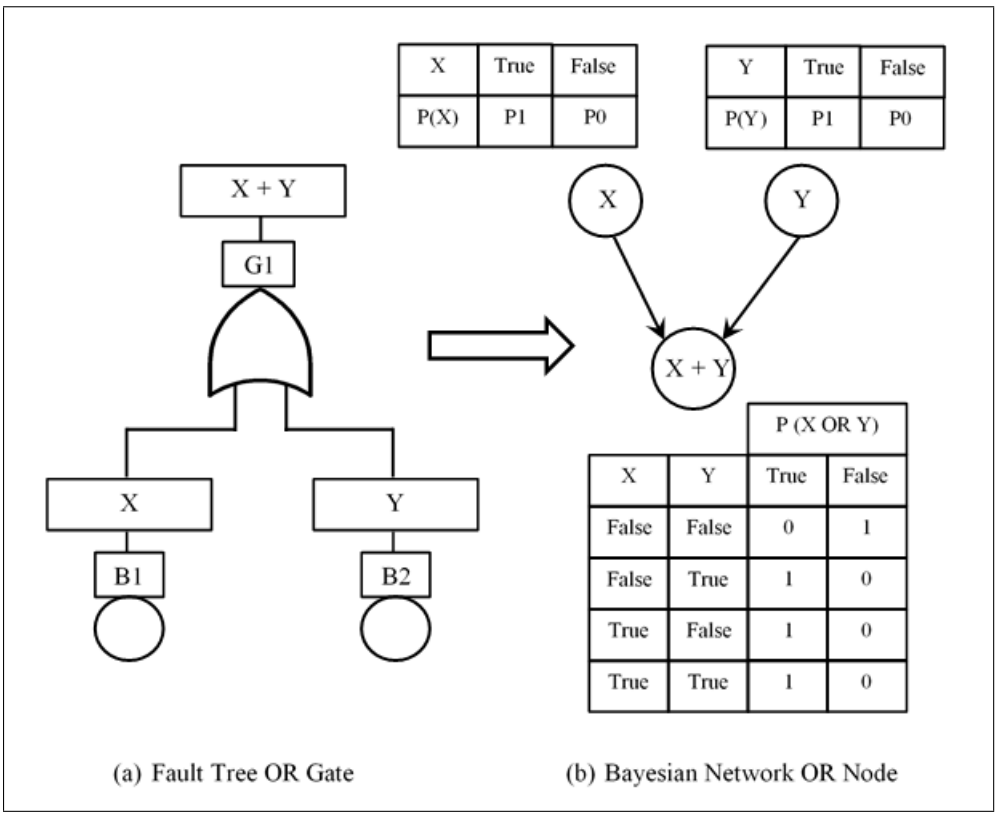


Figure 33: Two input OR gate and its BN model

According to the Royal Society for the Prevention of Accidents (RoSPA, 2011), human error is somehow involved in 95% of road accidents and in 76% of cases human error is solely responsible for accidents. Kim and Seong (2006) proposed a BN model which includes human factors to perform quantitative analysis of instrumentation and control (I&C) systems in nuclear power plants. Recently, to quantify human error probability, Jiang *et al.* (2010) have proposed a method which combines Cognitive reliability and an error analysis method (CREAM) (Hollnagel, 1998) with Bayesian Networks to overcome the limitations of general data from databases directly.

The popularity of BNs is increasing rapidly in performing dependability analysis of complex systems due to their ability to make decisions from uncertain situations. Researchers have already used BNs to evaluate popular reliability analysis techniques, e.g., FTs and DFTs; however, till now no effort has been made to use BNs for the evaluation of Pandora temporal fault trees.

2.5.3 Petri Nets

Petri Nets (PNs) are a formal graphical and mathematical modelling tool which are appropriate for specifying and analysing the behaviour of complex, distributed and concurrent systems (Murata, 1989; Peterson, 1977). They are very popular for their utilisation of established mathematical methods such as reachability tree analysis, deadlock analysis and other formal analysis methods and validation tools (Hei *et al.*, 2011; Reza *et al.*, 2009). A classical Petri Net is a bipartite directed graph represented graphically by:

- A finite set of places $P = \{P_1,\ P_2,\ \cdots,\ P_M\}$ which typically represent conditions within the system being modelled and they are denoted graphically as circles.
- A finite set of transitions $T = \{T_1,\ T_2,\ \cdots,\ T_N\}$ symbolized by bars which represent events occurring in the system that causes change in the conditions (Places) of the system.
- A finite set of directed edges or arcs F. Edges can only connect transitions to places or places to transitions such that $F \subseteq (T \times P) \cup (P \times T)$.

Input and output arcs are directed edges drawn from places to transitions and transitions to places respectively. Input places known as pre-conditions of a transition represent the conditions that need to be satisfied for the transition to be activated and connected to the transition through input arcs. On the other hand, output places are known as post-conditions of a transition which represent the outcome of the transition and are connected to the transition through output arcs. The black dots inside the places are known as tokens which represents whether a place is active or not. A specified number of tokens are needed to be available in the input places of a transition for the transition to be enabled and on firing/ triggering the transition a specified number of tokens are removed from each input places and a specified number of tokens are deposited to the output places of the transition.

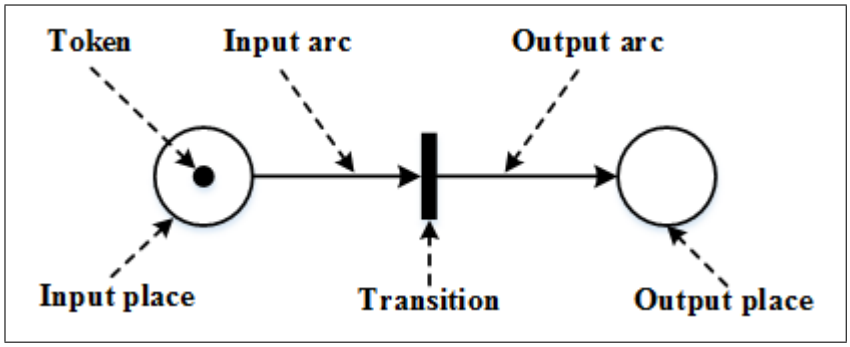


Figure 34: Basic components of a Petri Net

To increase the modelling capability of Petri nets, it has been extended with a special type of arc known as *inhibitor arcs*. Like other arcs, an inhibitor arc also connects a place to a transition and is represented by an arc terminating with a small circle instead of an arrowhead like normal arcs. In Figure 35, the arc from place $P2$ to transition $T1$ is an inhibitor arc. The interesting property of an inhibitor arc is that it enables the transition when the input place has no token and disables the transition when the place has a token (opposite behaviour of normal arcs). At the same time no tokens are consumed through the inhibitor arc. Therefore, the inclusion of the inhibitor arc in Petri nets allows for checking non-occurrence of events, and thus increases the modelling power of Petri Nets.

Marking functions which map tokens to places represents the dynamic feature of Petri nets. A marking of a PN at time t represents the state of the system at time t, and is

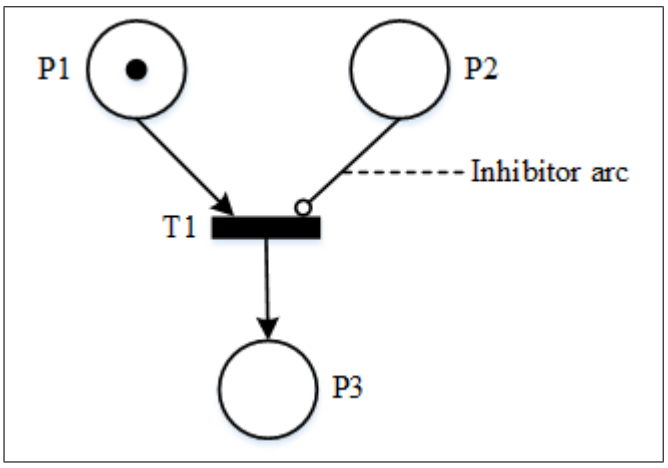


Figure 35: Petri Net with inhibitor arc

specified as a set of non-negative integers by:

$$M(t) = \{m_1(t),\ m_2(t),\ \cdots,\ m_N(t)\} \tag{16}$$

where $m_i(t)$ is the number of tokens in place i at time t and $M(t_0)$ is known as the initial marking.

A marking M_A is said to be reachable from another marking M_B if there exists a sequence of transitions from M_B to M_A. The reachability set of a PN is a set of all markings reachable from an initial marking $M(t_0)$.

Petri Nets capture the dynamic behaviour of a system by using a reachability tree/graph, and the process is known as reachability analysis. A reachability tree is a directed graph whose nodes are the possible markings of the PNs and edges represents the fired transitions. An edge exists from M_A to M_B if M_B is reachable via a single transition from M_A. Reachability can be analysed both forward and backward and the procedure is described by Leveson and Stolzy (1987), and Helmer *et al.* (2007).

For illustration purpose, we can consider a simple system with two identical repairable components and they can be either in an ON or an OFF state. A Petri Net of the system behaviour is shown in Figure 36. The Petri net of the system has two places named ON and OFF, and initially the system is in place ON which has two tokens (dots) i.e., both the components are active. Two possible transitions of the Petri net are denoted as $T_{failure}$ and T_{repair}. If the transition $T_{failure}$ fires then a token is removed from place ON and one token is deposited to place OFF, i.e., one component goes to the OFF state from the ON state. On the other hand, if transition T_{repair} fires then one token is deposited to place ON and one token is removed from place OFF, i.e., one component is recovered from its failure state and entered into its operational state.

Possible markings of the system can be defined as:

$\{2, 0\}$: both the components are active

$\{1, 1\}$: one component failed and another is working

$\{0, 2\}$: both the components failed (represents system failure)

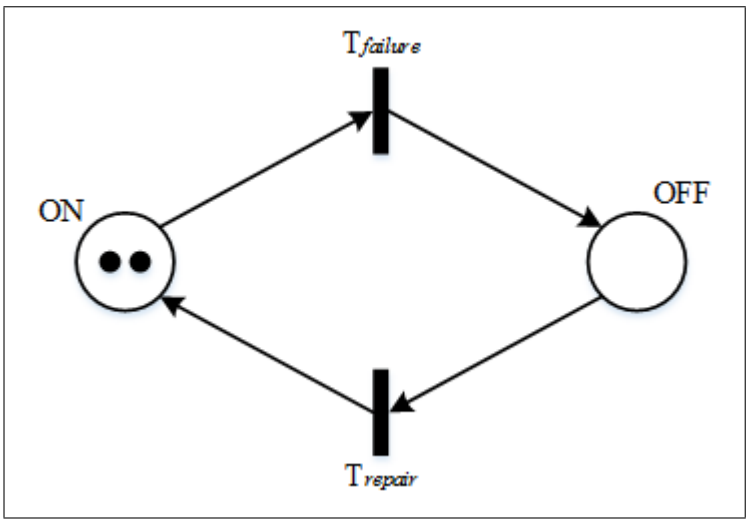


Figure 36: Petri Net of a system with two identical repairable components

From the Petri net of figure 36 and possible markings, a traceability graph can be drawn as shown in Figure 37. Therefore by using the forward reachability analysis feature of Petri Nets it is possible to show potential hazardous states of the system that can be reached from the initial state after a finite sequence of occurrence of events; it is believed that this feature would be helpful for this current research.

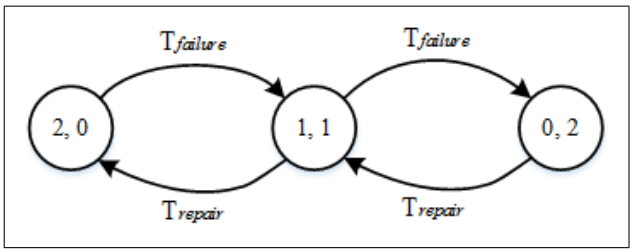


Figure 37: Traceability graph of the system with two identical repairable components

Classical PNs are easy to analyse but require more places and transitions to model the behaviour of moderately complex systems, which will give rise to state explosion problems (Desrochers *et al.*, 2005). Conventional PNs have gone through different modifications to overcome the problem of modelling data and time-dependent behaviour such as Arc Extensions and Inhibitor Arcs (Zurawski and Zhou, 1994), Timed Petri Nets, Coloured Petri Nets and Hierarchical Petri Nets (Desrochers *et al.*, 2005).

In Timed Petri nets, time is introduced to model the interaction among events considering their starting and completion times. A special case of timed Petri Nets is proposed by Molloy (1982) known as Stochastic Petri Net (SPN) where transition delays are exponentially distributed. One important feature of a SPN is that its underlying reachability graph is isomorphic to a continuous time Markov Chain (CTMC) (Marsan and Chiola, 1987). To allow both immediate and timed transition in the same Petri Net, SPNs are extended by Marsan *et al.* (1996) to introduce Generalized Stochastic Petri Nets (GSPN), where thin black bars are used to represent instant transitions and relatively thick bars are used to represent timed/delayed transition, and immediate transitions are given higher

priority than timed transitions. Coloured Petri Nets have been introduced to increase the modelling power of Petri Nets where tokens are given different colours to carry complex information (Jensen, 1996).

Petri Nets can be used to model both nominal and failure behaviour of systems. For instance, Fanti *et al.* (2014) have used Petri Nets to model an Integrated System (IS) for the Healthcare at Home (HAH) management system, and subsequently simulated the PN model to observe the behaviour of the IS. Some researchers have used Petri nets to model functional behaviour of the system and then use another safety analysis method e.g. FTA to perform the safety analysis based on the non-functional behaviour identified from the PNs (e.g. Reza *et al.* (2009)). Leveson and Stolzy (1987) introduced a new method where both nominal and failure behaviour of a system is shown in the same Petri net, i.e., failure transitions and fault conditions are introduced with normal transitions and normal conditions, and they are represented by double bar and double circle as shown in Figure 38.

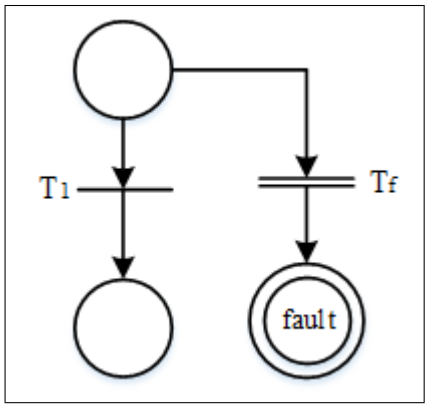


Figure 38: Petri Net showing nominal and failure behaviour

To increase the modelling and analytical power of combinatorial approaches of safety analysis, researcher like Bobbio *et al.* (1999), and Helmer *et al.* (2007) have proposed the ways of mapping FTs to PNs, and Codetta-Raiteri (2005) and Zhang *et al.* (2009) have introduced the ways of converting DFTs to PNs. Conversion of a Boolean Gate (OR) and a DFT's Gate (PAND) to PNs by Codetta-Raiteri (2005) is shown in Figures 39 and 40 respectively. The PAND gate is also available in the Pandora temporal fault trees with some behavioural changes. Therefore, the above transformation of PAND gate to PNs could be modified to make it Pandora compatible, and using these ideas other temporal gates of the Pandora could be transformed to Petri nets and thus benefit from the analytical power of Petri Nets.

State Machines provide an easy-to-use modelling approach, but they can be harder to analyse directly, therefore to take benefit of Petri Nets to analyse system behaviour, Pais *et al.* (2011) have presented the translation strategies from UML state machines to Petri Net and Hei *et al.* (2011) have presented an automated tool for transforming UML state charts to Petri Nets. But none of the methods has performed the equivalence check between the UML and Petri Net models. As state machine-based models suffer from

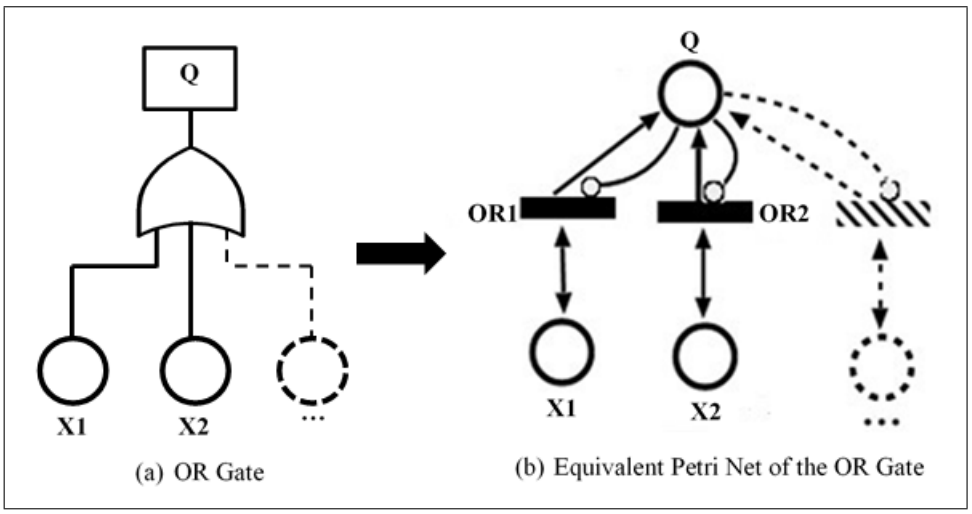


Figure 39: OR gate and its PN model

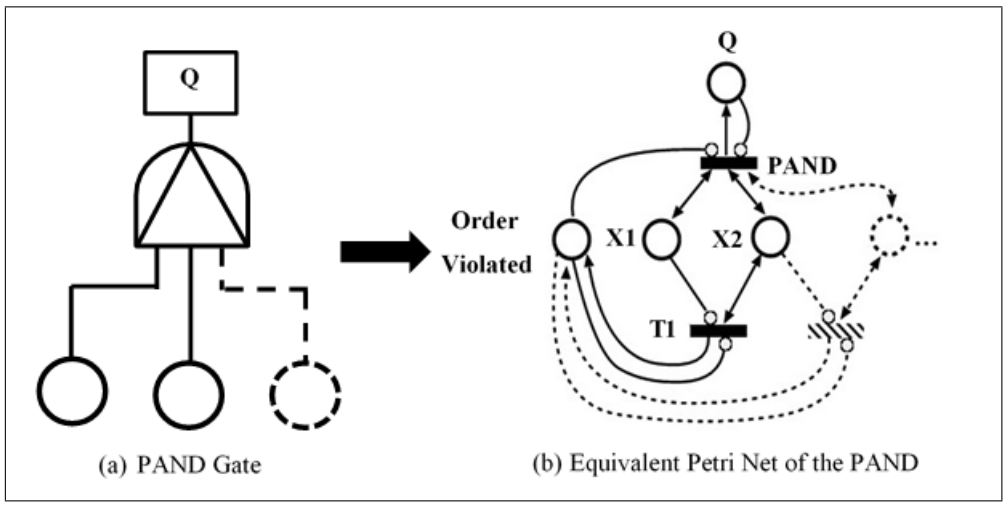


Figure 40: PAND gate and its PN model

state explosion problems and do not allow direct analysis, therefore we can benefit by converting those models to Petri Nets, and one important contribution would be to find the way to check equivalency between a state machine-based model and Petri Net of the same model.

2.6 Discussion

This chapter has presented a review of literature in reliability engineering, different classical safety assessment techniques, and in particular model-based safety assessment techniques. Model-based safety assessment is a continually evolving field that has attracted a significant interest in industry and academia over the last 20 years. Different tools and techniques have been developed as a part of model-based safety assessment to allow the analysts to perform different types of analyses automatically or semi-automatically. This chapter has classified different MBSA tools and techniques into two main categories based on their ways of working — *failure logic synthesis and analysis approaches* and

behavioural fault simulation approaches. However, irrespective of their structural complexity or whatever classical methods they use, all the techniques share a common goal: to provide some means to allow dependability analysis of safety-critical systems.

There exist many challenges in the MBSA area. One such problem is that of developing fault propagation models that links causes to effects. Another challenge is optimisation of system architectures to meet requirements or to allocate requirements. Hierarchically Performed Hazard Origin & Propagation Studies (HiP-HOPS) is one of the advanced and well supported state-of-the-art MBSA techniques which can resolve these challenges. It can provide similar functionality to CFTs and FPTN but with a higher degree of automation. Moreover, it can automatically generate fault trees, FMEA tables, perform quantitative analysis on fault trees, and has the ability to perform multi-objective optimisation of the system models.

Increasingly, systems are getting more complex and their behaviour becomes more dynamic. As the behaviour of the system changes, functions and their failure modes vary, as do the flows between components of the architecture and the potential deviations of those flows. Due to this complex behaviour and the many possible interactions between the components, assessing the effects of combinations of failure events is not enough by itself to capture the system failure behaviour. In addition, understanding the order in which they fail is also required for a more accurate failure model. However, classical approaches like fault trees (FTs) are not capable of capturing sequence-dependent dynamic behaviour. As these classical approaches are used as the primary basis for dependability analysis in many different MBSA techniques, those MBSA techniques also lack the capability of capturing dynamic behaviour of systems.

By looking at different state-event based approaches, it is clear that state machines make it easier to capture the dynamic behaviour of systems. For instance, Architecture Analysis and Description Language (AADL) with its Error Annex is an architecture modelling language which provides a rich syntax to annotate components of complex-systems with dependability related information based on stochastic state automata. AltaRica Dataflow and SAML are two other languages that also provide the ways of annotating components with dependability related information using a state-based formalism to capture dynamic behaviour of components. Papadopoulos (2000) has provided some preliminary ideas of using state-based annotation in HiP-HOPS to overcome its limitation in capturing dynamic behaviour, but no formal guidelines were provided to show how the annotations would be made or how the interactions between behavioural state machines of different components in different levels of the hierarchy will take place. Inspired by the capability of the state-based annotation languages to capture dynamic behaviour of systems, this thesis aims to develop a methodology to provide a way to annotate system components with dynamic behaviour related information using state-event automata, thus allowing the compositional MBSA techniques to perform dynamic dependability analysis based on these annotations. Considering the multi-dimensional contributions of HiP-

HOPS to the MBSA domain, it is selected as an example MBSA tool to illustrate the use of the newly developed methodology in practice.

As already seen in this chapter, the main limitation of the state machine based approaches is that for a moderately complex system they suffer from state explosion and at the same time it is not possible to analyse the state machines directly. As it is possible to arrange system architectures hierarchically, it is therefore believed that the state explosion problem could be minimised by using hierarchical state machines (state charts). However, proper guidelines on hierarchical system modelling, e.g. how the communications between the states in different levels of the hierarchy would take place, need to be provided. By foreseeing the complexities that may arise while using state machines for dynamic system modelling, this thesis will provide guidelines on hierarchical system modelling using state machines to alleviate the complexities.

As it is not possible to readily analyse state machines directly, we therefore also need to translate state machines to other representations, e.g. Pandora temporal FTs or BNs. Some researchers have shown the ways of translating state machines to classical fault trees but it will not be able to preserve the sequence dependent behaviour of the system. However, recently, Mahmud *et al.* (2012) have introduced the ways of translating state machines into Pandora temporal fault trees. Therefore, if system components can be annotated with state-based behaviour, then Pandora temporal fault trees can be generated from the state machines to show the failure behaviour of the system.

Pandora allows fault trees to capture sequence-dependent dynamic behaviour. The major advantages of Pandora are that by performing qualitative analysis it can create useful insight to system failure in the absence of quantitative data and it is integrated well in model-based design and analysis. Although Pandora's strength lies in its ability to perform qualitative analysis, it has limited capability of performing quantitative analysis. However, in order to obtain accurate information about the system dependability, it is necessary to perform quantitative analysis along with the qualitative analysis. Quantitative analysis provides measures of system reliability probabilistically and also the criticality of the components based on their relative contribution to the occurrence of system failure.

Considering the increasing importance of model-based design and analysis, and the potential use of Pandora in this context, recent efforts have been made to develop analytical and Monte Carlo Simulation based approaches to allow quantitative analysis of Pandora TFTs. However, the application of the analytical solution is limited to exponentially distributed data only; on the other hand, the Monte Carlo Simulation based approach is too computationally expensive for larger systems. Therefore, there exists plenty of scope to explore different approaches to develop novel methodologies to provide an improved solution to quantitative analysis of Pandora TFTs. The new methodologies should be able to provide a state-space solution to Pandora TFTs in a reasonable time, and at the same time, the methodologies should be equally applicable to both exponentially and non-exponentially distributed data.

With an aim to provide methodologies to provide improved analysis of Pandora TFTs, different approaches such as Markov Chains (MCs), Bayesian Networks (BNs), and Petri Nets (PNs) have been studied. All these approaches have established a good reputation in the dependability analysis domain. Markov Chains are a popular probabilistic model suitable for dependability assessment of dynamic systems and dynamic fault trees (DFTs) are usually quantified by translating them to MCs. But to show dynamic behaviour of complex systems it requires a large number of variables, and thus suffers from state explosion problem. Moreover, the application of MCs is limited to exponentially distributed data only.

Stochastic Petri Nets (SPNs) are well-established and a powerful mathematical modelling tool. As their underlying reachability graphs are isomorphic to Continuous Time Markov Chains, they have been used to provide a state-space solution to DFTs. PNs offer several important advantages such as the graphical feature of PNs allows effective modular modelling and its mathematical feature allows model verification and simulation. However, like the MCs, their application limited only to exponentially distributed data.

Bayesian networks, another popular tool which provides a robust probabilistic method of reasoning under uncertainty, is increasingly used to assess dependability of systems. It can be observed from the literature review that BNs are increasingly being used to provide solutions to DFTs. Unlike MCs and PNs, they are capable of analysing systems featuring different failure distributions. The major limitations of BNs are that there are no formal semantic guidelines for developing BNs for a system and they do not guarantee a coherent model. To guarantee a coherent model, one option would be to transform another coherent classical model to BNs, e.g. FT into BNs, but the success in this case is limited by the correctness of the classical model from where BNs is obtained. Another option would be to define rules to develop BNs which could generate coherent model of system.

According to the above discussion, BNs appear to be the most suitable techniques to provide a state-space solution to Pandora TFTs which will be applicable to both exponentially and non-exponentially distributed data. Considering the mathematical strength and many other advantages offered by PNs, these also seem to be a good candidate technique that could be explored to provide a state-space solution to Pandora TFTs. Therefore, these two techniques will be considered in this thesis for the quantitative analysis of Pandora.

Another important issue in the MBSA area that has received very little attention is the uncertainty in the failure data. The outcome of a quantitative analysis is entirely dependent on the accuracy of the failure data or other numerical values used in the analysis. In most of the existing MBSA techniques, failure data of components are considered to be always available and thus do not consider inclusion of a degree of uncertainty in the failure data of the basic events. As already discussed in this chapter, for many complex systems, it is often very difficult to estimate a precise failure data or probability of components from past occurrences due to lack of knowledge (vagueness), scarcity of statistical data, and changes in operating environments of the systems etc.(He *et al.*, 2007; Mah-

mood *et al.*, 2013; Tanaka *et al.*, 1983). From the literature review, it is clear that the fuzzy logic has the capability to deal with linguistic variables and it provides an efficient way to draw conclusions from imprecise and vague information. Fuzzy logic is widely used to handle uncertainty in FTA but no effort has been made to use fuzzy logic to handle uncertainty in Pandora temporal fault trees. For this reason, fuzzy logic has been selected as a potential way of incorporating uncertainty aspects of failure data in the quantification process of Pandora TFTs. This thesis will explore the existing quantification techniques for the Pandora TFTs to develop a methodology to allow quantitative analysis of Pandora with uncertain data.

Chapter 3

Dynamic Dependability Analysis

3.1 Introduction

One important aspect of many systems is their dynamic behaviour, i.e., the behaviour of the system (both nominal and potential failure behaviour) can change according to what state or mode of operation the system is in. As the behaviour of the system changes, functions and their failure modes vary, as do the flows between components of the architecture and the potential deviations of those flows. These behavioural uncertainties are increasingly prevalent as the complexity and scale of the systems increases, and dynamic behaviour can be observed in almost any large industrial systems. Dynamic behaviours of systems create complicated situations for classical dependability analysis techniques. The first aim of this chapter is to discuss those complications. The second aim is to show a potential way of modelling dynamic behaviour. The third aim is to use that behavioural annotation in the context of HiP-HOPS to perform dynamic dependability analysis.

3.2 Dynamic Behaviour and Challenges in Dependability Analysis

In large industrial plants, big tasks are often divided into smaller tasks and are processed in different stages of the operation. In this way, resources are utilised in a sequence of different stages, and in each of those stages, a set of different functions are performed to complete the overall task. The Aircraft Fuelling Systems (AFS) of modern aircraft are examples of such systems. The operation of those systems can be divided into modes, whereby some of the operations may take place before the flight and some may take place during the flight. Throughout the process, at any particular point in time, some of the system components may act as active components and some others may act as passive components. By active at a point in time, we mean those components which are engaged in some sort of useful system operation at that particular time. On the other hand, passive components are those which are idle or switched off, i.e., not involved in any sort of

operation at that point in time and waiting to be reactivated by the system.

The activity and inactivity of system components could be intentional or unintentional. Sometime a system may have to perform a set of variable functions, and to facilitate this sort of varying functionality, a variable configuration of the system is obtained by intentionally activating and deactivating a selected number of components. A second scenario could be that a system is performing a fixed set of functions, and in the presence of the failure of one or more of the system components, the system may sacrifice some of its non-critical functions and go to a degraded operational mode by only doing the critical functions with a limited number of components with a different configuration. To make the safety critical system more tolerant to faults, many systems have fault tolerance strategies built in. As part of such a fault tolerant strategy, in the presence of faults, systems may reconfigure themselves by using spare (cold or hot) components to respond to the faults and continue the nominal behaviour. A component is considered as cold spare if it is deactivated until it is required in any sort of system operation. On the other hand, hot spare components are always active irrespective of their necessity in the system operation; however, they serve their functionality only when the primary component fails.

For instance, consider a hypothetical twin engine aircraft fuel distribution system, shown in Figure 41. The system has two fuel tanks, TL and TR; two pumps PL and PR; and three valves VL, VR, and VC. In the normal operating condition, mode 1, the left engine receives fuel from the left tank (TL) through VL and PL; and the right engine receives fuel from the right tank (TR) through VR and PR. In this mode, valve VC is kept closed (see Figure 42). Now, if we hypothesise a failure such that VR is stuck closed, then there is no flow from TR to the right engine. So the system has to reconfigure itself to maintain fuel flow to the right engine. A new configuration of the system in this mode, mode 2, is shown in Figure 43. And a third configuration, mode 3 (see Figure 44), is formed when there is no fuel flow to the left tank from TL through VL.

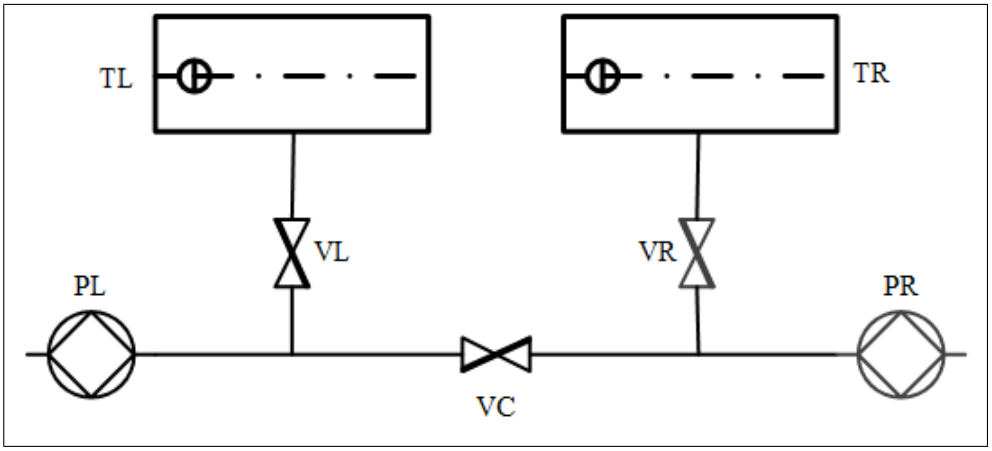


Figure 41: Twin engine aircraft fuel system

This kind of multi-modal operation and variable number of system configurations can pose difficulties for classical dependability analysis techniques, e.g. fault trees, and for those model-based safety assessment techniques that use these classical techniques as a

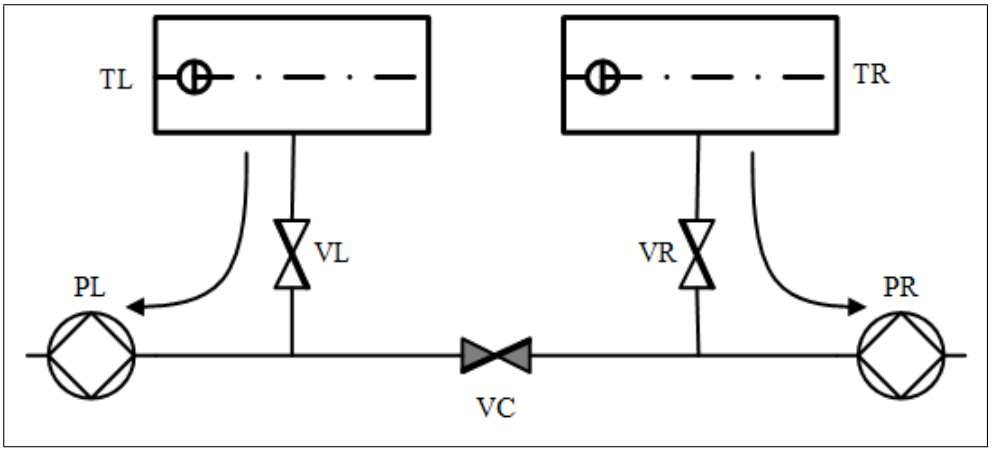


Figure 42: Direction of fuel flow to engines in Mode 1

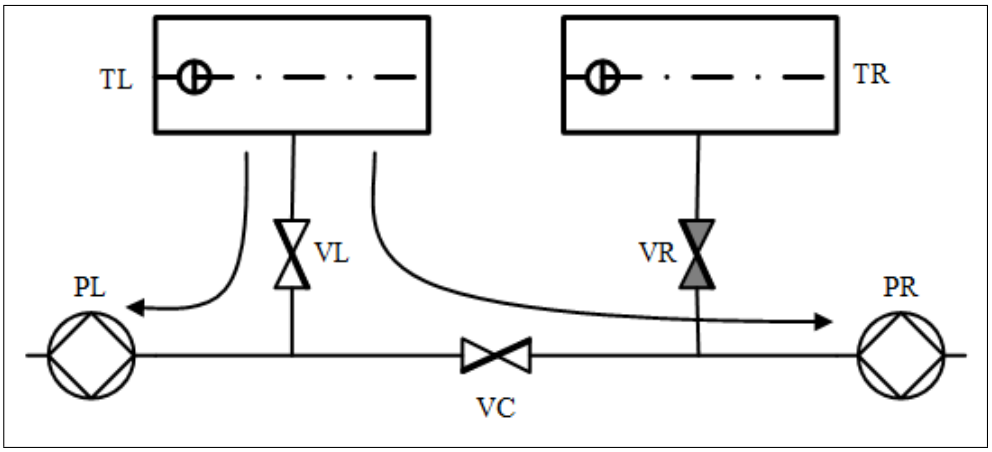


Figure 43: Direction of fuel flow to engines in Mode 2

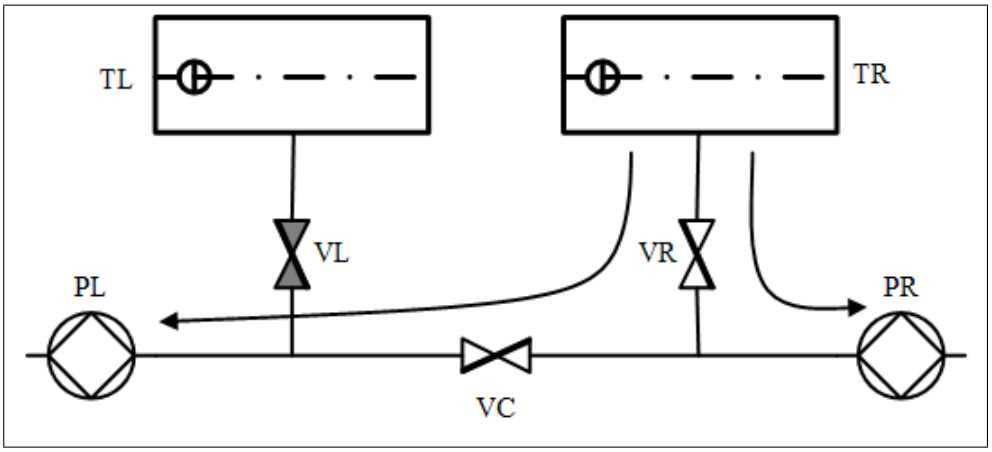


Figure 44: Direction of fuel flow to engines in Mode 3

means for their analysis. If we want to analyse such a system with tools like HiP-HOPS, we will soon be faced with difficulties caused by the dynamic behaviour of the system. For such systems, it is difficult to precisely define the nominal behaviour of the system because it has different behaviour in different modes. Therefore, it is equally difficult to define the potential deviations from the nominal behaviour. Another thing to note is that different selections of components are activated and deactivated to obtain a desirable configuration; therefore some of the components may be irrelevant in some of the modes, and thus so are their failure modes. For example, if the above mentioned system is in mode 2 then the valve VR becomes irrelevant because it is either failed or has no functionality in this mode. As a result, it is a challenge for classical analysis technique to take this state dependent behaviour into account and represent it in an understandable and manageable format.

Some extensions of classical fault trees have been proposed to accommodate state/mode dependent dynamic behaviour in the structure of the fault tree. The Dynamic Fault Tree is one such extension that proposes four new logic gates: the functional dependency gate, the spare gate, the priority-AND gate, and the sequence-enforcing gate. However, the primary focus of this extension is to perform quantitative analysis, and this is done usually by translating them into Markov chains. Therefore, it gives the analysts limited scope to obtain qualitative information about the complex dependencies between modes and the potential failure behaviour of the components. Pandora is another extension of classical fault trees that captures dynamic behaviour using three temporal gates: priority-AND, priority-OR, and simultaneous-AND gates. It also defines a set of temporal rules to allow meaningful qualitative analysis. However, methods for quantitative evaluation of Pandora fault trees are not well developed yet. Although all the dynamic extensions of the classical fault trees provide some means to capture the dynamic behaviour of systems, there still remains the issue of correctly modelling the dynamic behaviour and linking it to dynamic analysable models that can be used for qualitative and quantitative analysis.

Furthermore, modelling the mode/state dependent dynamic behaviour complicates the dependability analysis process and makes the analysis results inconsistent. We believe that if the components can be annotated with failure behaviour based on the modes of operation they are involved in, and if this information can be organised in a consistent and manageable way and put in the correct context, then the complexity of the dynamic analysis can be minimised and the analysis can be made more consistent. Here, we can foresee two potential challenges. The first challenge is to represent the mode automata, i.e., what is causing the system to go from one mode to another and how the transitions are happening. The second challenge is how we can associate the modes with the failure behaviour of the components and thus what could be the best way to represent that information. In both challenges, the goal is to find representations compatible with common MBSA approaches.

In the following sections we describe an approach to represent the mode-based be-

haviour of dynamic systems and ways of annotating system components with mode based failure behaviour. The first step involves identifying potential modes of operation for the system and then representing the transitions between modes with a suitable structure. In the second step, potential failure behaviours of the system components in different modes are identified, the components are annotated with those behaviours, and all the data are finally presented in a well-structured format. At the end of the chapter, an example dependability analysis is performed by using HiP-HOPS to demonstrate the approach's utility with MBSA techniques.

3.3 Representation of Dynamic Behaviour

According to the above description, a system can behave dynamically by reconfiguring itself to provide a variable set of functions or by providing a single set of function with some alternative configurations. In both cases, the configuration of the system has a link to the functions provided by the system. In other words, the architecture of the system has a relation with the mode in which the system is operating, i.e., a distinct architecture can be associated with a distinct mode of operation. This section presents a way of representing the mode/state based behaviour of systems as well as a way of associating modes with the failure behaviour of components.

3.3.1 Representing Dynamic Behaviour with Mode Charts

State-transition based methodologies, such as Mode Charts, are suitable and widely used to capture dynamic behaviour of systems. A mode chart M could be defined as (Košecká, 1992; Sampath *et al.*, 1996):

$$M = (Q, \Sigma, \delta, q_0) \tag{17}$$

where Q is the set of all possible modes, Σ is the set of all possible events, δ is the transition function $\delta : Q \times \Sigma \rightarrow Q$ and q_0 is the initial mode. Example of a mode chart is shown in Figure 45.

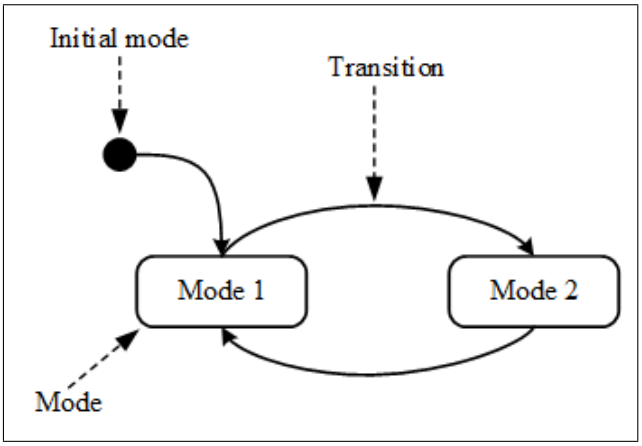


Figure 45: Example of a mode chart

In this thesis, dynamic behaviour of a system is represented as a set of different functional modes and a set of transitions among the modes. In each of these modes, the system may deliver a fixed set of functions with one architecture or may deliver a different set of functions with different architectures. For example, a system can be in:

- a *normal operational mode*, in which it delivers a set of functions;
- a *degraded mode*, where the system is operating safely but with limited functionality;
- a *failed mode*, where all the intended functionality is lost.

However, the variations in system modes and their interactions depend on the designers' intention, i.e., how they want the system to behave in different situations. Therefore, to describe the dynamic behaviour using mode charts, we would have to identify all possible functional modes that the system could be in and also need to determine all possible conditions that can trigger transitions between those functional modes.

Consider, for instance, the fuel distribution system presented earlier (see Figure 41), where the system has to perform a fixed set of functions. Let's say the functions are:

1. To provide fuel to the left engine.
2. To provide fuel to the right engine.

As seen from Figure 42 to 44, to provide the above functions the system could work in three different modes. Figure 46 shows a mode chart of the twin engine aircraft's fuel system considering three possible functional modes. Each of the modes essentially represents a distinct configuration of the system. The association between modes and different configurations of the system is shown in Figure 47. Mode 1 is the initial mode, and in this mode all the system components are working fine: the first function of the system is completed by providing fuel to the left engine from the left tank through VL and PL, and the second function is obtained by directing fuel to the right engine from the right tank through VR and PR. Mode 2 and 3 refers to two degraded modes where system functionality is not affected, but the system nonetheless changes its configuration in response to failure.

When the system is in Mode 1, it will remain in this mode unless transition T1 or T2 causes it to go to Mode 2 or Mode 3. In Mode 2, fuel flow to right engine from right tank is replaced by the fuel flow from the left tank, whereas in Mode 3, fuel flow to the left engine from the left tank is replaced by the fuel flow from the right tank. Transition conditions could be a single event (normal or failure) or a combination of events and they could be represented as Boolean expressions.

Let us find the expression for T1, as seen from Figure 47, that causes the system to change its mode of operation from the Mode 1 to the Mode 2 because of the lack of flow to the right engine through the valve VR. This deviation could be caused either due to

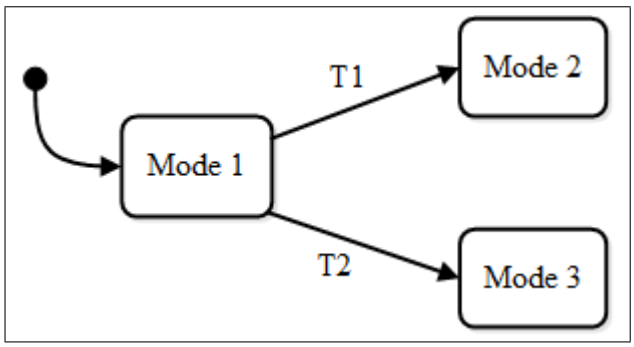


Figure 46: Mode Chart of the behaviour of the system in Figure 41

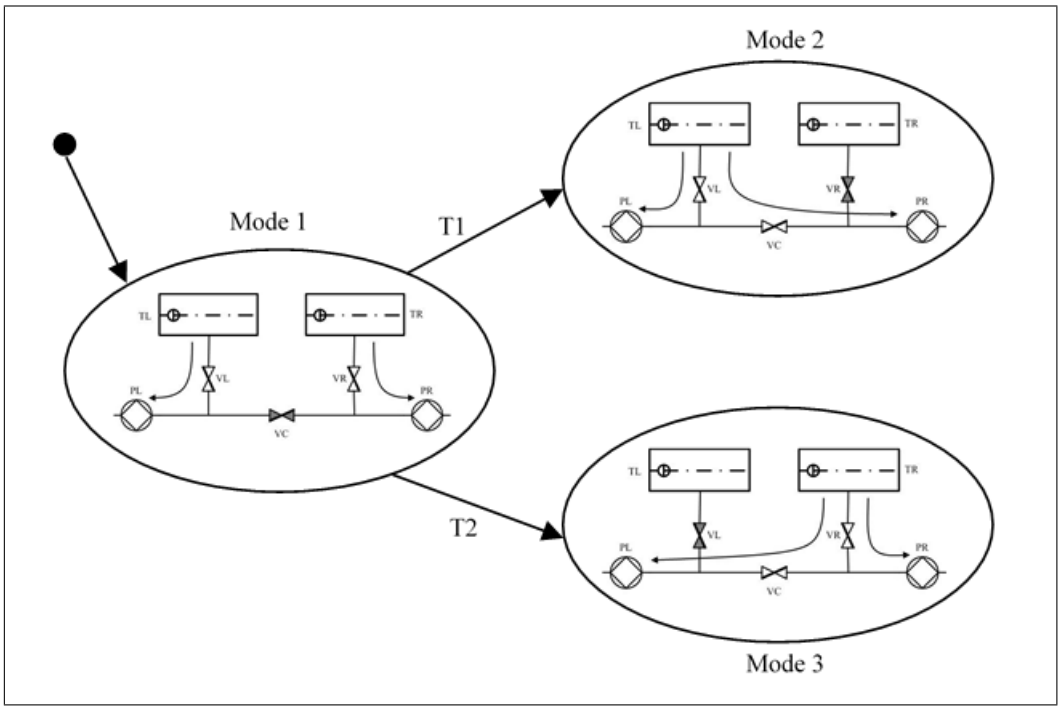


Figure 47: Association between system architectures and modes

the failure of tank TR or failure of valve VR. So the expression of the transition could be written as:

```
T1 = Failure of TR \/ Failure of VR
```

The two terms in the right hand side of the above expression are the failure of two of the system components. The above expression is used only for illustrative purpose, however, and this expression can be represented in a better format, as will be seen in the next subsection. Each component may be active in some modes and inactive in others, and therefore the failure behaviour of each component is confined to those modes in which it is active. If a particular component is not involved in any kind of operation in a mode, then the behaviour of that component may become irrelevant for that mode. Figure 48 shows a slightly different version of the mode chart of Figure 46, where all the components involved in each mode are listed inside the rounded rectangle representing the mode and the transitions are represented as failure events.

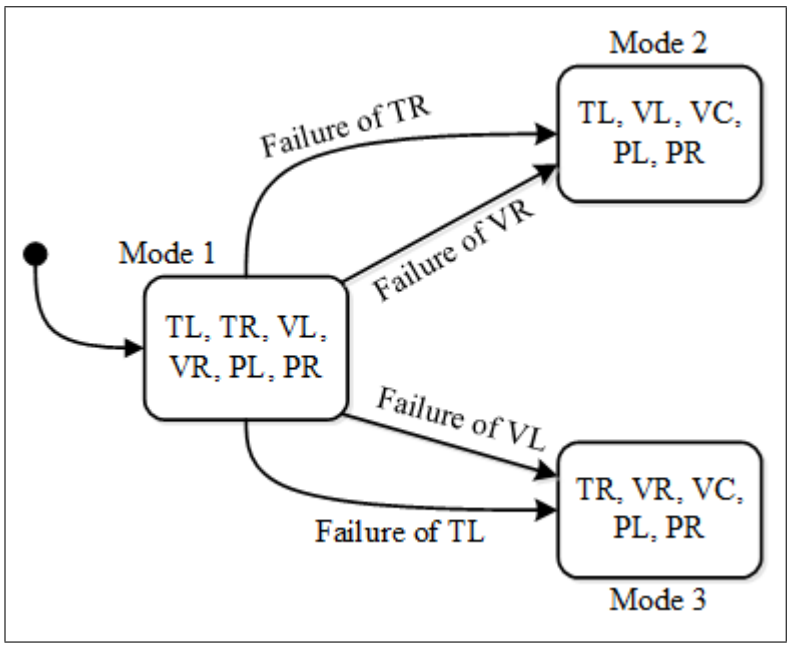


Figure 48: Mode chart with active components listed in the modes

3.3.2 Annotation of Mode Based Failure Behaviour of Components

The mode chart of a system only represents the *functional* modes and/or *degraded* modes and all possible transitions between those modes. However, in order to perform dependability analysis, we need to identify the *failed* modes of the system as well. As mentioned earlier, the system can work with different configurations in different *functional* modes to provide the same or different set of functions; therefore, a single *failed* mode could be reached in different ways from different functional modes. This is the primary bottleneck for the classical dependability analysis approaches.

In compositional MBSA approaches, system components are usually annotated with a fixed set of failure behaviour; however, for systems with dynamic behaviour, this kind of annotation is often misleading, either too optimistic or too pessimistic. For dynamic systems, components are instead required to be annotated with mode based behaviour, i.e., which mode the components are in and what the possible behaviours are for that mode. It is not necessary for all components to have failure behaviour in all modes. This is due to the fact that a component may not be active in a particular mode, may be failed prior to entering a mode, or the activity of a functioning component is masked due to the failure of other components. However, this assumption is only appropriate for non-repairable components. If a component is repairable then a failed component can regain its functionality to change the configuration of the system, and thus make the above assumption invalid.

Let us deal with the various cases one at a time. In the first case, where a component is not active in a particular mode, the failure behaviour of that component is irrelevant in this mode. For example, in the above mode chart, component VC has no activity in Mode 1; hence, we can say that the failure behaviour of VC is irrelevant in Mode 1.

In the second case, the component has already failed before coming to the present mode; therefore, the failure behaviour of the component is already addressed in any of the prior modes. For example, assume that the system goes from Mode 1 to Mode 2 due to the failure of TR (see Figure 48). If we look at Figure 48, then we can see that TR was present in Mode 1 but not in Mode 2. Therefore the failure behaviour of TR is addressed in Mode 1 before entering Mode 2, and in Mode 2 it becomes irrelevant.

In the third case, the component itself is not failed; however, its activity does not have any effect in the system because of some other reason, e.g., failure of other components. For example, if some components are connected in series, then failure of one component makes the subsequent components inaccessible. This situation can be seen more clearly from the example above. Assume that the system in above example goes to Mode 2 due to the failure of TR. Failure of TR makes the activity of VR ignorable in Mode 2, because the activity of VR is only relevant when TR is working properly.

Following the above discussion, the mode based failure behaviour of component VL, i.e., the possible causes of its failure, can be represented as follows:

Table 7: Mode based failure behaviour of component

Failure Behaviour / **Component Name**	**Mode 1**	**Mode 2**	**Mode 3**
VL	VL.Blocked	VL.Blocked	No effect

In Table 7, for illustrative purposes, we assume that only a blockage can cause the failure of valve VL; however, any other causes (or combination of causes) can be used. It is worth noting that the failure behaviour of VL is the same in Mode 1 and Mode 2; as VL is not active in Mode 3, VL's failure behaviour has no effect in this mode.

Let us consider the inability of the above system to deliver fuel to the left engine as one of its failure modes. Transitions to this *failure* mode from all three *functional* modes are shown in Figure 49. The failure mode is represented as a dotted ellipse and the transitions that cause the system to go from a *functional* mode to a *failure* mode are represented as dotted arrows. For illustration, the mode based failure behaviour — in the form of input and output deviations — for two failure modes are shown in Table 8. As seen in the table, in mode 1, the condition 'no fuel to the left engine' could be caused by the failure of the left pump (PL.Failed); however, in mode 2 and 3, the same failure could be caused because of:

- no flow from valve VL to PL (No_VL_PL) in disjunction with the failure of PL
- and no flow from valve VC to PL (NO_VC_PL) in disjunction with the failure of PL respectively.

No_VC_PL is only possible in mode 3 because in mode 1, VC is in passive mode, thus no flow is possible through VC, and in mode 2, there is a flow through VC, but the flow

is in the opposite direction, i.e., VC to PR, hence flow from VC to PL does not make any sense in this mode. And in mode 3, `No_VC_PL` could be caused by the failure of VC (`VC.Blocked`) or because of no flow from VR to VC (`No_VR_VC`).

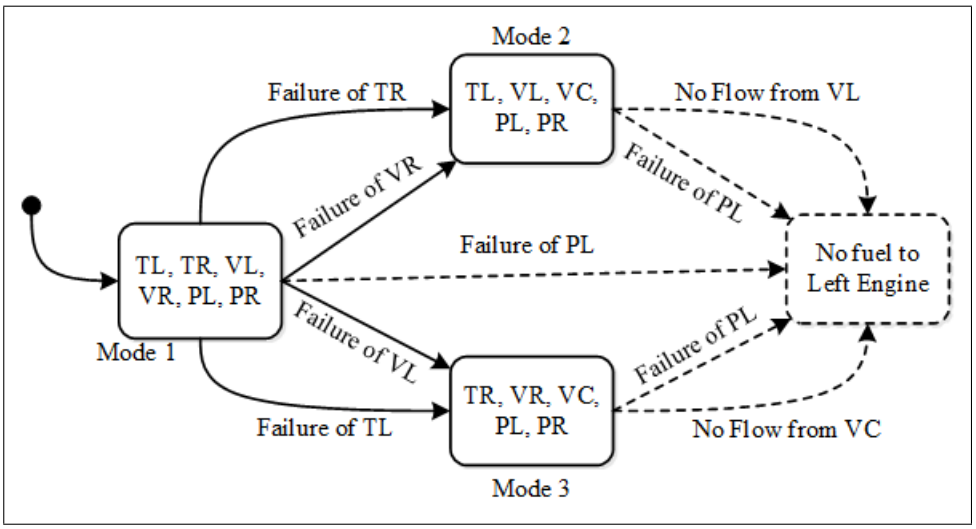


Figure 49: Mode chart with failure modes and transitions to failure modes from functional modes

Table 8: Mode based expressions of failure behaviour

Failure Expression / Failure Modes	Mode 1	Mode 2	Mode 3
No fuel to left engine	PL.Failed	PL.Failed ∨ No_VL_PL	PL.Failed ∨ No_VC_PL
No_VC_PL	No effect	No effect	VC.Blocked ∨ No_VR_VC

3.3.3 Complexity of Mode Chart and Hierarchical Modelling

In the last two subsections, we have seen the mode chart of the functional behaviour of the system and the mode based failure behaviour of system components. The architecture of the example system was relatively simple; hence the mode chart was fairly small. Now the question is whether these approaches can be applied to model the dynamic behaviour of large and complex systems. Such systems usually perform a wide range of functions in different phases of operation is often composed from multiple smaller subsystems. As a result, the architecture of the system can become too complex to model in a straightforward fashion because of the many subsystems and many possible interactions among them.

If we want to model the behaviour of such a system using the concept shown in the previous subsection, then we would soon be faced with serious challenges. First, we have to identify all possible functional modes and the related configurations of the system. After that, all permutations of all the modes needs to be calculated and then all possible transitions between those modes are required to be defined. In this process, many redundant modes and repeated transitions will be considered. Finally, we have to compile all the information into a single model, resulting in a very big and complex mode chart which could easily be unmanageable. This problem is often referred as the state explosion problem.

This problem could be solved by introducing a hierarchical structure to the mode chart and thus make the modelling of dynamic behaviour of complex system more manageable. This section will therefore show how a hierarchical mode chart can be used to model the dynamic behaviour of a more complex system.

The most important difference between hierarchical mode charts and classical mode charts is that the hierarchical mode charts introduce hierarchically nested modes. Modes that contain other modes are called composite/super modes; on the other hand, modes without internal structure are called basic/simple modes. A nested mode is called a direct sub-mode when it is not contained by any other mode; otherwise, it is referred to as a transitively nested sub-mode.

Every mode can react to external events for transition, and while reacting to external events, every mode can have optional entry and exit actions. When a mode is entered, the entry actions are executed and during leaving a mode the exit actions are executed without considering how a mode is entered or left. Every mode should contain the information about the components of the system involved in that mode. If the mode is a super mode then it should have a default starting sub-mode, which acts as a target mode of a transition made to the super mode, and should have at least one final state to handle outgoing transitions.

Modes can have any number of mode variables, which are local to the modes and are used to reflect the status of the components involved in the modes. If a variable belongs to a super (parent) mode or if any behaviour is defined for a parent mode, then all the sub-modes of that super mode will inherit that variable or behaviour, i.e., all sub-modes use it as a global variable/behaviour.

Every sub-mode must continue to act as though it was also an instance of the super mode, which means no sub-mode should act in a way which contradicts/violates the aims of the super mode. The best possible way to ensure this is to control the interaction of every sub-mode with modes outside the parent modes. To do so, some restrictions are put on the scope of interaction for modes. All basic modes which do not belong to a super mode can communicate with each other at any level of the hierarchy and can also communicate with super modes that are at the same level in the hierarchy. Any kind of sub-modes within a super mode can interact with each other directly because they are at

the same level, but they are not allowed to interact with any modes outside of their super mode directly even if they are at the same level. If any sub-mode has to interact with other modes outside of the super mode, then the interaction should take place under the control of the super mode. That means all transitions internal to a super mode are handled by its sub-modes, but any outgoing or incoming transition to or from any outside mode is handled only by the super mode itself. In other words, we can say that the super/parent mode has responsibility for any transitions external to that mode.

So far, we have said that any transitions to or from external modes are handled by the parent modes. The following rules are followed to handle any kind of transition in our proposed hierarchical mode chart.

Incoming Transitions to a Parent Mode: If a transition is made to a parent mode, then any entry action for that parent mode is performed first. After that, the transition is handled by the parent mode by transferring the control to the default starting sub-mode.

Incoming Transitions to a Sub-Mode of a Super Mode: If a transition is made to a sub-mode of a super mode, then any entry action for the enclosing super mode is performed first. After that, the control is transferred to the targeted sub-mode based on the transition condition.

Outgoing Transitions from a Super Mode: As mentioned earlier, every super mode has at least one final mode. If the final mode is reached by a transition from the super mode itself, then this final mode represents a transition from the super mode. A transition from the super mode means transition from any of the sub-modes because of inheritance and takes precedence over any of the transitions from the current sub-mode. This is referred as "common behaviour" shared by all sub-modes.

Outgoing Transitions from a Sub-Mode: If a final mode of a super mode is reached by a transition from one or more of the sub-modes, then this final mode represents a transition from the sub-mode of the super mode to any external mode. In such case, any exit action of the sub-mode is executed followed by any exit action of the enclosing super mode.

To illustrate the use of hierarchical mode charts to capture the dynamic behaviour of complex systems, consider a modified version of the system from Figure 41, as shown in Figure 50. Some new components have been added to new architecture to make it fault tolerant. Now both the tanks have two openings and each of the openings is connected to a valve. For example, for the left tank (TL), left opening (TLO1) is connected to VL1 and right opening (TLO2) is connected to VL2. So, fuel to the left engine from TL can be directed through two different paths. The first path is through TLO1 $\rightarrow$ VL1 $\rightarrow$PL (left path), and the second path is TLO2 $\rightarrow$ VL2 $\rightarrow$ PL (right path). Similarly, for the right

engine, fuel from the right tank (TR) could flow through either TRO1 → VR1 → PR (left path) or TRO2 → VR2 → PR (right path).

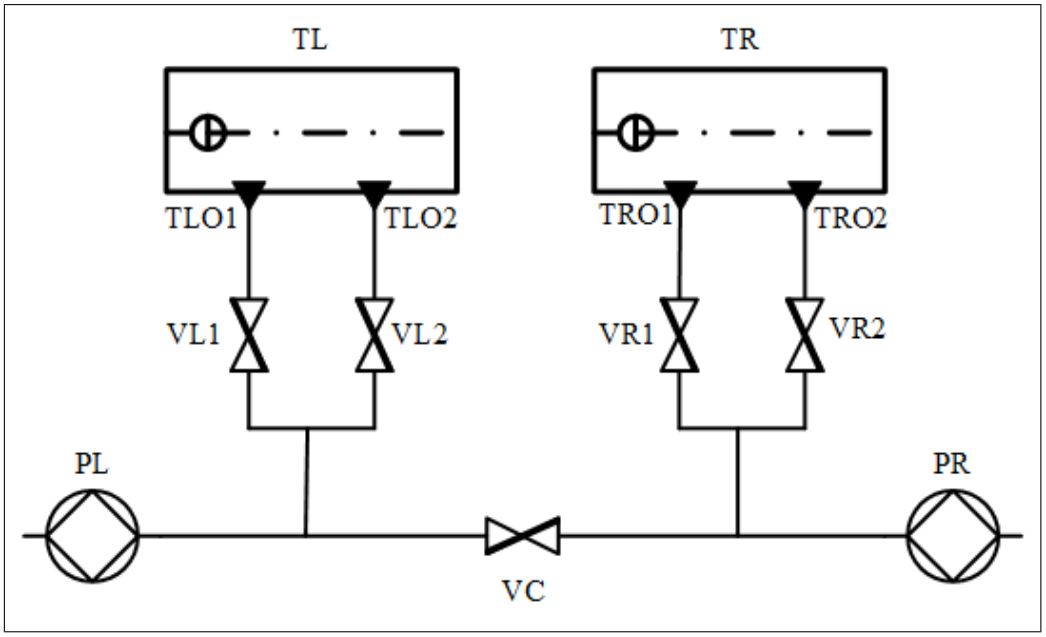


Figure 50: Fault tolerant twin engine aircraft fuel system

Consider that the behaviour of the system is defined as such that at the beginning both the tanks use the left path to provide fuel to the respective engines, and the right paths stay inactive, i.e., components in the right paths are acting as passive components. If for some reason the left lines become unavailable, then the tanks will switch to the right lines to continue the fuel flow to the engines. After that, if the right path of the left tank becomes unavailable (fails) then the system will continue its operation using the right tank; on the other hand, if the right path of the right tank becomes unavailable then the system will depend only on the left tank for fuel flow to both the engines.

Now, let us consider that the failure of the system to provide fuel to the left engine is one of the failure modes (hazardous state). The flat mode chart showing this failure behaviour is shown in Figure 51. In this mode chart, the name of the modes are shortened to increase readability; however, for better understanding Table 9 and 10 provide elaboration for the shortened name of the modes with detailed description of the modes with reference to the system status. For this particular failure mode, *no fuel to left engine*, there are eight *functional* and/or *degraded* modes which are represented as black rounded rectangles in the mode chart. In these modes, proper fuel flow to the left engine is ensured, but each time with different system configurations. For example, the initial mode, **`TL_L_TR_L`**, represents a *functional* mode where the left engine receives fuel from the left tank through its left path and the right engine is fed from the right tank through its left path. On the other hand, the **`TR_L`** is a *degraded* mode and it represents a scenario where both the engines are fed from the right tank through its left path.

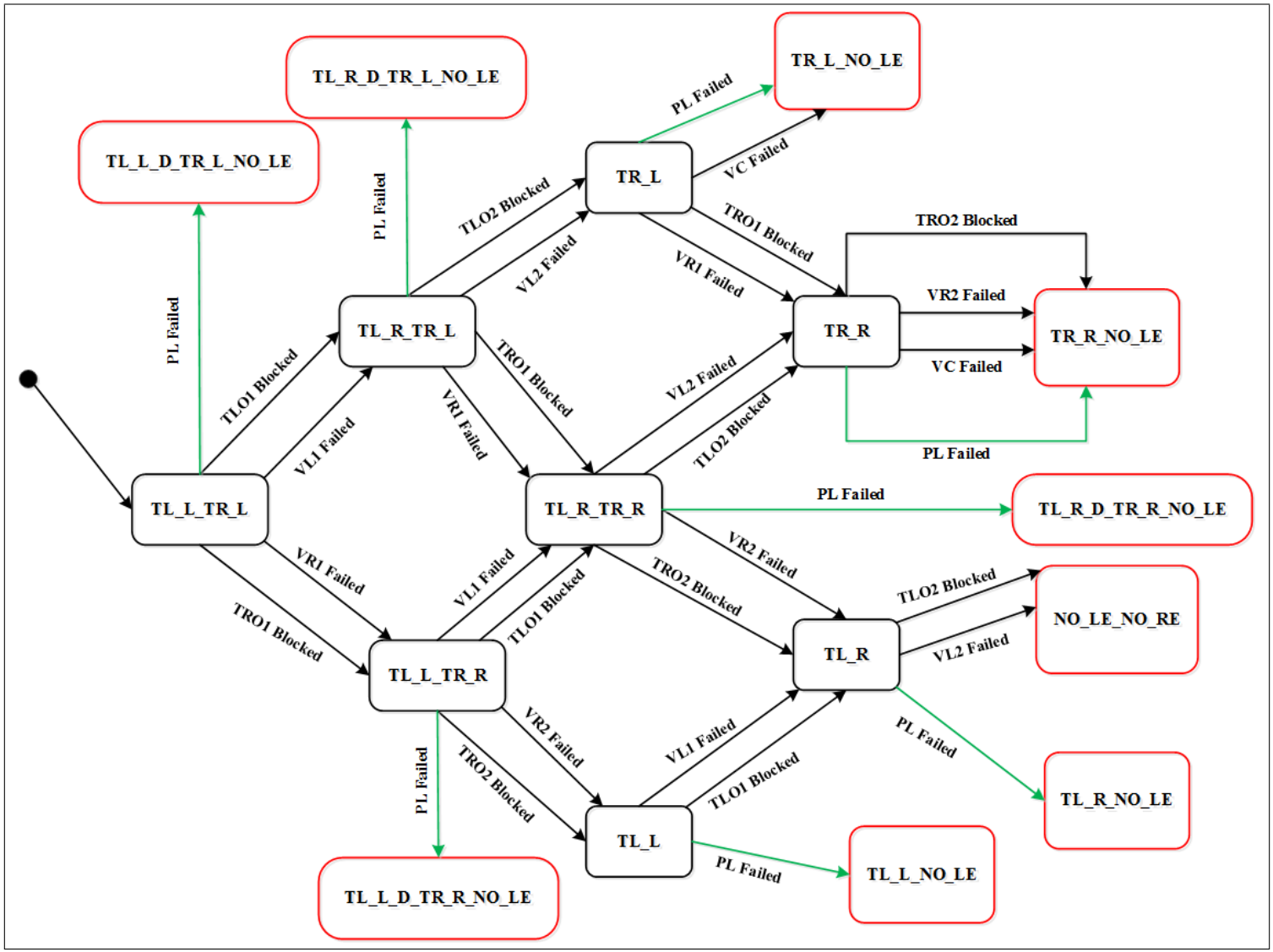


Figure 51: Classical Mode Chart for the failure behaviour ‘no fuel to left engine’ of the system in Figure 50

Table 9: Explanation of the name of the *functional/degraded* modes used in the mode chart of Figure 51

Mode Name: Shortened	**Mode Name: Explained**	**Description of the Mode**
TL_L_TR_L	TL is using its left path and TR is using its left path	The left tank (TL) is providing fuel to the left engine and the right tank (TR) is providing fuel to the right engine. To provide fuel to the engines, both the tanks are using their left path.
TL_L_TR_R	TL is using its left path and TR is using its right path	TL is providing fuel to the left engine through its left path and TR is providing fuel to the right engine through its right path.
TL_R_TR_L	TL is using its right path and TR is using its left path	TL is providing fuel to the left engine through its right path and TR is providing fuel to the right engine through its left path.
TL_R_TR_R	TL is using its right path and TR is using its right path	TL is providing fuel to the left engine through its right path and TR is providing fuel to the right engine through its right path.
TL_L	TL is using its left path	TL is providing fuel to both the engines using its left path.
TL_R	TL is using its right path	TL is providing fuel to both the engines using its right path.
TR_L	TR is using its left path	TR is providing fuel to both the engines using its left path.
TR_R	TR is using its right path	TR is providing fuel to both the engines using its right path.

Table 10: Explanation of the name of the *failed* modes used in the mode chart of Figure 51

Mode Name: Shortened	**Mode Name: Explained**	**Description of the Mode**
TL_L_NO_LE	No fuel to the left engine and TL is using its left path	Fuel flow to the left engine is stopped and the left tank (TL) is providing fuel to the right engine using its left path.
TL_R_NO_LE	No fuel to the left engine and TL is using its right path	Fuel flow to the left engine is stopped and TL is providing fuel to the right engine using its right path.
TR_L_NO_LE	No fuel to the left engine and TR is using its left path	Fuel flow to the left engine is stopped and TR is providing fuel to the right engine using its left path.
TR_R_NO_LE	No fuel to the left engine and TR is using its right path	Fuel flow to the left engine is stopped and TR is providing fuel to the right engine using its right path.
TL_L_D_TR_L_NO_LE	No fuel to the left engine, TL is using its left path but is in dormant mode and TR is using its left path	No fuel flow to the left engine is available. The left path of the TL is available for supplying fuel to the right engine but currently it is not in use because the right engine is getting fuel from TR through its left path.
TL_L_D_TR_R_NO_LE	No fuel to the left engine, TL is using its left path but is in dormant mode and TR is using its right path	No fuel flow to the left engine is available. The left path of the TL is available for supplying fuel to the right engine but currently it is not in use because the right engine is getting fuel from TR through its right path.

TL_R_D_TR_L_NO_LE	No fuel to the left engine, TL is using its right path but is in dormant mode and TR is using its left path	No fuel flow to the left engine is available. The right path of the TL is available for supplying fuel to the right engine but currently it is not in use because the right engine is getting fuel from TR through its left path.
TL_R_D_TR_R_NO_LE	No fuel to the left engine, TL is using its right path but is in dormant mode and TR is using its right path	No fuel flow to the left engine is available. The right path of the TL is available for supplying fuel to the right engine but currently it is not in use because the right engine is getting fuel from TR through its right path.
NO_LE_NO_RE	No fuel to the left and the right engine	Fuel flow to both the engines is stopped.

The *failed* mode could be reached from each of the *functional* and *degraded* modes, with the same or different transitions. In the mode chart, there are nine modes (red rectangles) that represent the same *failed* mode, *'no fuel to left engine'*. Although, these modes represent the same failure behaviour they have different significance from the system's perspective. For example, modes **TL_L_NO_LE** and **TL_L_D_TR_R_NO_LE** both correspond to the system state where there is no fuel to the left engine and the left path of the TL is available to provide fuel to the right engine. But still they are distinctive in the sense that in the former mode the left path of the TL is already in use to provide fuel to the right engine because fuel flow to the right engine from TR is no longer available. On the other hand, in the latter mode, although the left path of TL is available to provide fuel to the right engine, it is not in use because fuel flow to the right engine from TR is still available through the right path of TR. Therefore, although in both the modes there is no fuel flow to the left engine, with respect to overall system performance and in terms of criticality, the mode **TL_L_NO_LE** is more critical than the mode **TL_L_D_TR_R_NO_LE**. This is because in the latter mode, in the condition of failure, more options would be available for the fuel flow to the right engine. More specifically, in **TL_L_NO_LE** mode, if the left path of the TL becomes unavailable then the system will only have the option to switch to the right path of the TL.

On the other hand, in the **TL_L_D_TR_R_NO_LE** mode, if the right path of the TR (currently in use) becomes unavailable then the system will switch to the left path of the TL. That means the system will enter the **TL_L_NO_LE** mode. Therefore, for the failure mode "no fuel to the left engine", both the modes have same significance — but for the overall system operation (in this case, fuel flow to the remaining engine, i.e., right engine) their significance is different.

For this reason, a single failure behaviour is represented in different modes. If we observe the mode chart carefully, then we can see that the transition **PL Failed** (marked as green arrow) causes the system to go to the failure mode irrespective of the *functional/degraded* mode the system is in. Therefore, this transition can be considered as a common transition which repeats eight times, and the repetition can be omitted by using the hierarchical mode chart.

To create a hierarchical mode chart, we need to create parent modes and identify the modes from the flat mode chart that can be assigned to a single parent mode. For the mode chart of Figure 51, we can create parent modes based on which tanks are in use in the system operation. The hierarchical mode chart of the functional behaviour of this system is shown in Figure 52. As seen in Figure 52, three parent modes — **TLTR**, **TL**, and **TR** are created. The **TLTR** mode represents a scenario when both the tanks are in use, i.e., the left engine gets fuel from the left tank and the right engine gets fuel from the right tank.

It is seen from Table 9 that each of the modes **TL_L_TR_L**, **TL_R_TR_L**, **TL_L_TR_R** and **TL_R_TR_R** correspond to the scenarios when the system is using both the tanks. Hence, these four modes are included as sub-modes in the parent mode **TLTR**. The **TL** mode represents the system behaviour when both the engines get fuel from the left tank. The sub-modes to the **TL** mode are **TL_L** and **TL_R**, taken from the flat mode chart. And finally the **TR** mode represents the scenario where both engines get fuel from the right tank and the sub-modes to this mode are **TR_L** and **TR_R**.

At first look, it may seem like that the top-level functional behaviour of this system is same as the system in Figure 41; however, it is not. For the system in Figure 41, in mode 1, both the tanks have a single path available to provide fuel to the engines, hence, unavailability of one path would force the system to go to a different *functional* mode. But for this system, every tank has two possible paths to use; therefore, failure of the first path (left path) would not cause the system to go directly to another *functional* mode. Therefore, in **TLTR** mode the system can operate with different configurations and each of the configurations is considered as a sub-mode in the parent mode, **TLTR**.

As seen in Figure 52, the **TLTR** mode has four sub-modes, and **TL_L_TR_L** is the default starting sub-mode, therefore any transition targeted to **TLTR** mode would lead to a transition to the **TL_L_TR_L** sub-mode. This sub-mode represents a fully functional configuration of the system where both the left and the right tank are using their left path to provide fuel to the left and the right engine respectively.

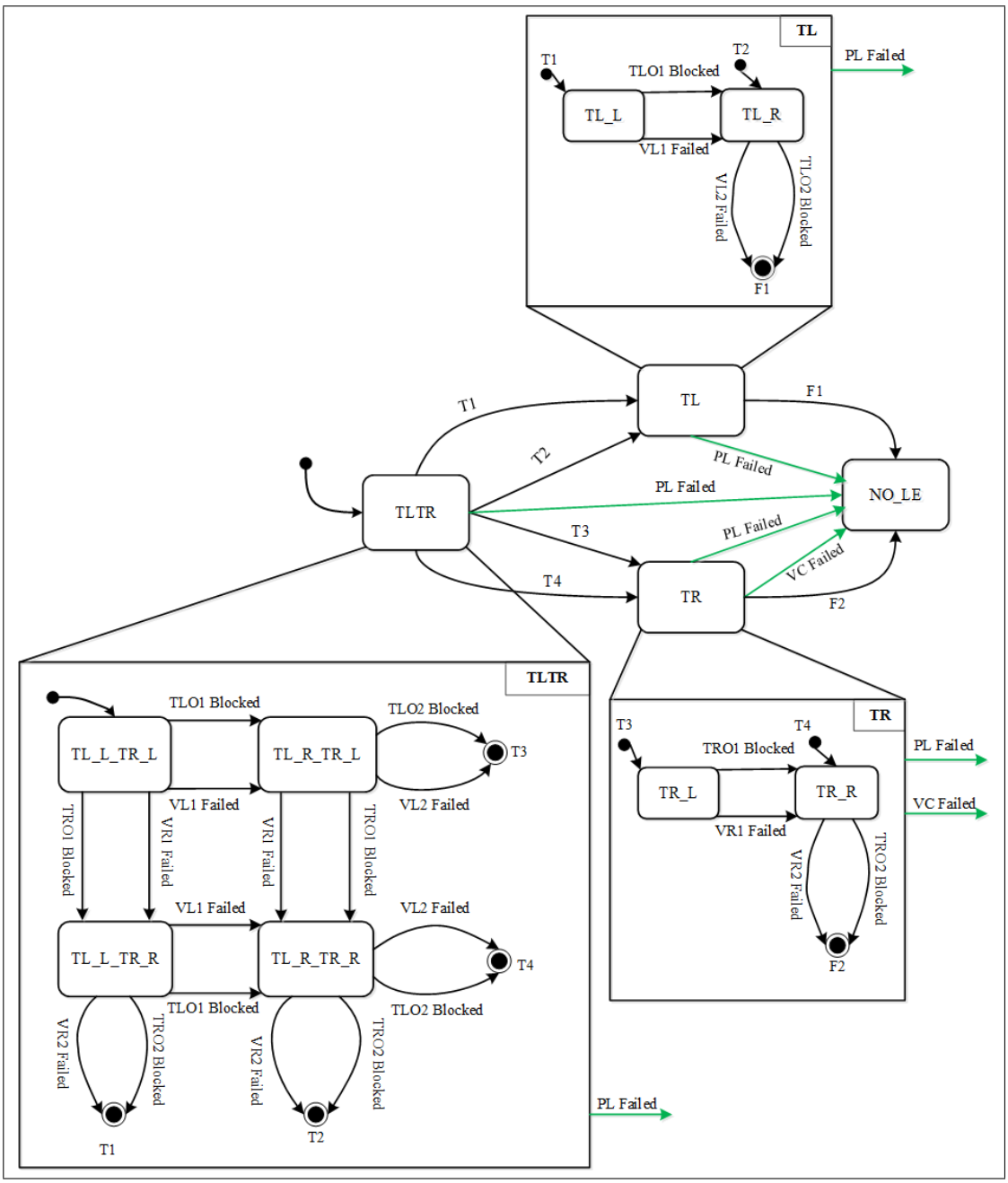


Figure 52: Hierarchical structure of the mode chart of Figure 51

The **TL_R_TR_L** sub-mode corresponds to a configuration where the left engine is using the right path and the right tank is using the left path. In this mode, the left tank is left with only the right path and therefore the failure of this path would cause the system to leave the **TL_R_TR_L** sub-mode, and consequently leaves the **TLTR** mode. As this transition is from a sub-mode of the **TLTR** mode to an external mode (**TR**), the transition is handled by a final mode within the **TLTR** mode, shown as transition **T3** leading to a final state (a filled circle within a circle). Similar to the **TL_R_TR_L** sub-mode, in the **TL_L_TR_R** sub-mode the right tank is left with its right path only, therefore unavailability of this path would also cause the system to go to the **TL** mode. Finally, the **TL_R_TR_R** sub-mode corresponds to a configuration where both tanks have already lost their left path and only have their right path in action, hence the failure of any of the paths

would force the system to go either to the **TL** or the **TR** mode.

Before (see Figure 51) the transition **PL Failed** was defined for all the *functional* or *degraded* modes that causes the system to go to a *failure* mode, but now that common transition is defined only for the parent modes (depicted by the green arrows in Figure 52). As the transition is defined for the parent modes, all the sub-modes will therefore automatically inherit this behaviour. **PL Failed** is defined at the level of the **TLTR** mode and causes the system to go to the **NO_LE** mode (no fuel to the left engine), therefore, the occurrence of **PL Failed** in any of the sub-modes of **TLTR** will cause the system to go to the **NO_LE** mode. In the same way, **PL Failed** is defined as a common transition in the **TL** and **TR** mode. In addition to **PL Failed**, the **PR** mode has also defined **VC Failed** as another common transition, because in the **TR** mode, this transition will take the system into the **NO_LE** mode no matter what **TR** sub-mode it is in.

3.3.4 Application of temporal logic to mode charts

Now, we will see when we need to take the order of the transitions/events into account while defining the behaviour of a system. In the mode chart, if a single mode is reached from another mode through two different paths with two or more different transitions in two different orders, then the order of transitions has no effect, i.e., the behaviour can be expressed as a Boolean expression. For example, in figure 52, inside the **TLTR** mode, the **TL_R_TR_R** sub-mode can be reached from **TL_L_TR_L** through two different paths. One of the paths contains the combination of transitions (**TLO1 Blocked** $\vee$ **VL1 Failed**) then (**TRO1 Blocked** $\vee$ **VR1 Failed**), and the other path contains the transitions (**TRO1 Blocked** $\vee$ **VR1 Failed**) then (**TLO1 Blocked** $\vee$ **VL1 Failed**). That means both the paths contain the same combination of transitions in two different orders. Therefore, in this case, the order of events has no effect on the behaviour, and we can write the expression for the paths as (**TRO1 Blocked** $\vee$ **VR1 Failed**) **AND** (**TLO1 Blocked** $\vee$ **VL1 Failed**) without using any temporal operators. However, this is not necessarily the case for all systems. In the mode chart of a system, if two different modes can be reached from a single mode with two or more transitions in a different order, then the order of transitions must be taken into account.

Consider the simplified railway crossing system in Figure 53. The system consists of a sensor, a barrier, and a warning light. When there is no train on the railway line, the barrier stays raised and the warning light is turned off. When a train approaches, the sensor sends a signal to the light and the barrier. On receiving the signal the light starts flashing and after a set amount of time the barrier descends. When the train has passed, the sensor sends another signal to the barrier and the light, and the barrier will raise and the light will stop flashing after receiving the signal from the sensor. The sensor can behave abnormally either by committing a signal when the signal is not required or by omitting a signal when the signal is required. If the sensor commits a signal when there is no train on the railway line, then the barrier will descend and the light will flash unnecessarily which will disrupt

the traffic flow. However, if the sensor commits a signal when a train is approaching and the barrier is already descended and the light is flashing, then the barrier will raise and the light will stop flashing, thus creating a hazardous scenario.

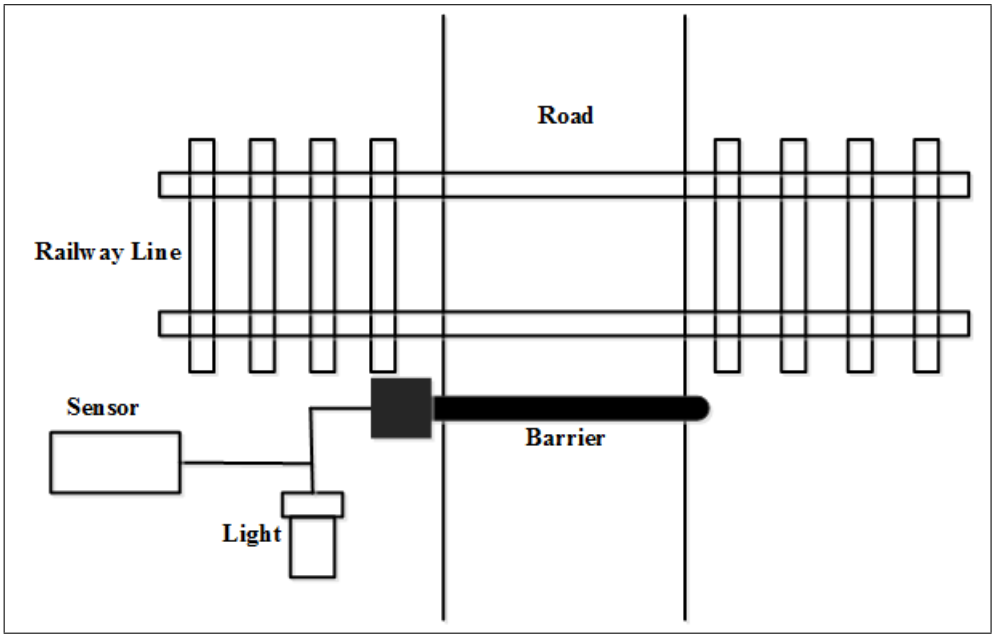


Figure 53: A simple railway crossing system

Now we come to the effects of the omission of the signal from the sensor. If a train is approaching and the sensor fails to send a signal to the light and the barrier then it will create a catastrophic scenario. In this scenario, the train will continue to cross the road and the drivers of the vehicles on the road will receive no warning about this because of the absence of the light and the barrier. As a result, it is highly likely that some of the vehicles may collide with the train. However, an omission of signal when the barrier is already descended and the light was already on has no such catastrophic consequences. In this case, the train will pass, but the traffic will be delayed because the barrier will stay descended and the light will stay on due to lack of signal. These are some of the scenarios that can be caused either by the commission or omission of signal. However, the barrier and the light themselves can be failed to cause some serious scenarios. If the light is blown then it is considered as the failure of the light and if the motor of the barrier fails, then it is considered as the failure of the barrier. It is clear that if the barrier motor fails when it was in the raised position, then even if the signal is made available by the sensor the barrier cannot descend. Similarly, if the light is blown then it cannot flash in the presence of the signal. The mode chart of the railway crossing system is shown in Figure 54.

In each of the modes we can see the position and status of the barrier, the status of the light, and whether the train is approaching or not. The barrier could be either raised (Barrier Up) or descended (Barrier Down) and in both the position it could be either OK or Failed. If the barrier is OK, then it can go from one position to another, otherwise it

will stay in its current position forever. In the mode chart, mode **M1** and **M4** represent two *functional* modes of the system and all other modes are either *degraded* or *failed* modes.

In **M1**, there is no approaching train, the barrier is in the up position (Barrier Up), and the light is off. Now, if a train approaches, then a signal (transition **Signal_LON_BDOWN**) will cause the system to leave the mode **M1** to go to the mode **M4**. On the other hand, when the train has passed, another signal (transition **Signal_LOFF_BUP**) will cause the system to go the mode **M1** by leaving mode **M4**. Now, let us investigate the effect of the order of the transition in the system behaviour, i.e., when we should take the order of events/transitions into account during defining the system behaviour. Assume the system is in mode **M1**: now the transition **Motor Failed** will cause the system to go to the mode **M7**. After that, the transition **Signal_LON_BDOWN** will take the system to the mode **M11**. The mode **M11** represents a scenario when the train is approaching, the light is flashing, but the barrier is stuck in the raised (Up) position. This scenario can be considered as serious because the train is crossing the road, but the barrier has not descended to stop the vehicles on the road. If the drivers of the vehicles on the road somehow missed the flashing light, then it is highly likely that they could collide with the train.
Now, let see what mode the opposite sequence of the transitions will take the system to and how serious that mode is. In mode **M1**, if the transition **Signal_LON_BDOWN** occurs first, then the system will go to the mode **M4**. After that, if **Motor Failed** transition occurs, then it will take the system to the mode **M9**. This mode represents a scenario when a train is crossing the road and the barrier is already descended and failed. That means when the train has passed, the barrier cannot raise to let the vehicle on the road to pass, thus causing unnecessary disruption in the traffic flow. However, this scenario is not as serious as in mode **M11**.

So, it is obvious that from a single mode a system can go to two different modes with different levels of severity just because of two transitions occurring in two different orders. Therefore, we can conclude that it is necessary for some systems to take the order of transitions into account and for some systems it is not required. This requirement completely depends on the architecture of the systems and how the components are interacting in the architecture. Generally, if different sequences of events will lead to different modes and have different effects, temporal operators are necessary. Therefore, we would suggest using both Boolean and temporal operators to define the failure behaviour of systems.

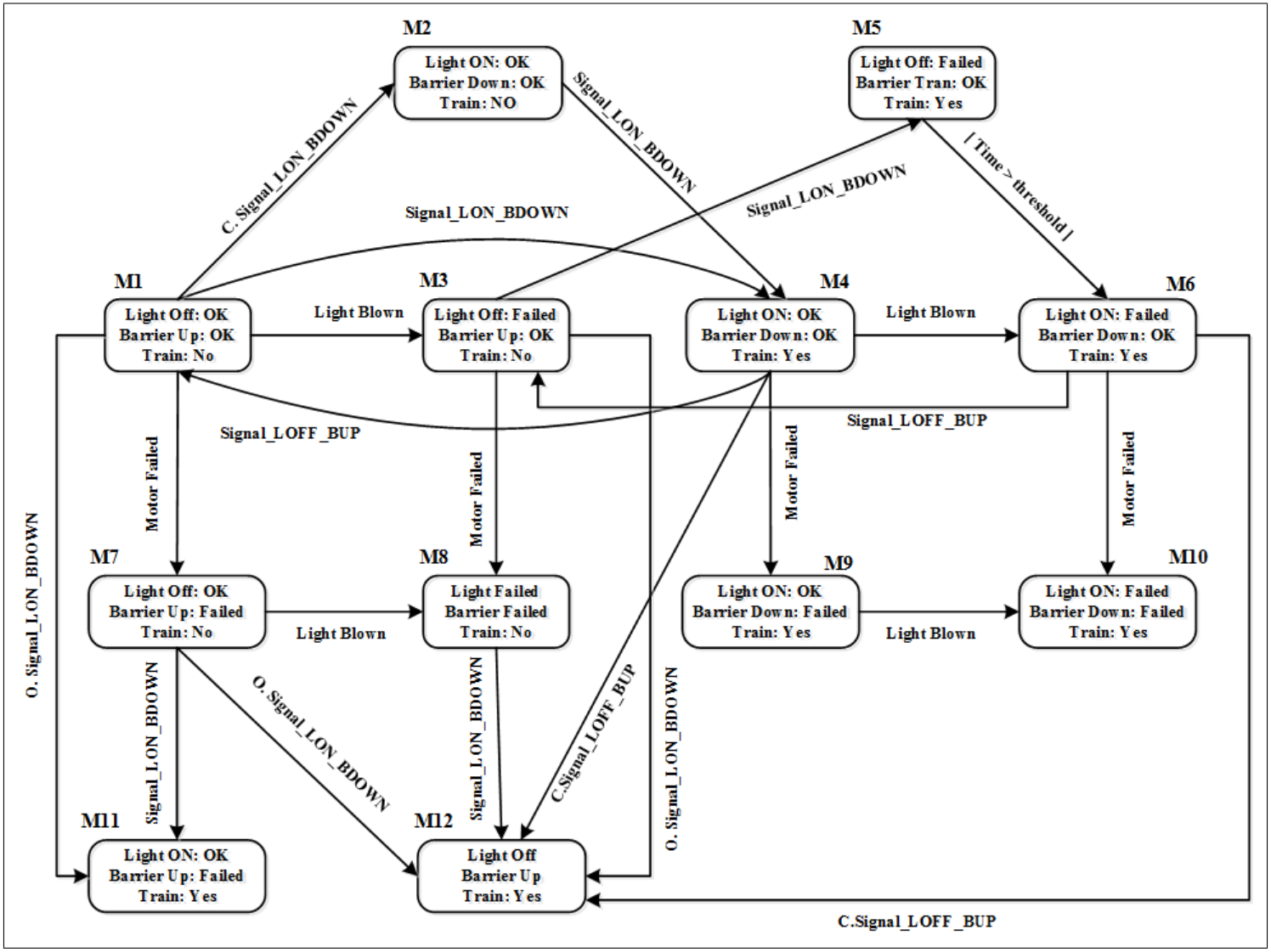


Figure 54: Mode chart of a simplified railway crossing system

3.4 Modelling and Dependability Analysis Process using HiP-HOPS

As described in section 2.3.5, HiP-HOPS analysis consists of three main phases. In the first phase, the system model is annotated with the failure data. The system model showing the topology of the system, components, and the interconnections between them is created in a separate modelling tool (e.g., the Matlab Simulink tool has an integrated interface for HiP-HOPS). We refer to this model of the system as a static architectural model and it can be hierarchical to facilitate the decomposition of the system into subsystems and so on (see Figure 55).

Annotation of system components with failure data consists of defining the various failure behaviours of the components, such as what can go wrong with a component and how the component will respond to the failures that have occurred elsewhere in the system. The behaviours are defined as Boolean expressions with output deviations as the operands in the Boolean operator. The output deviations include the unexpected omission of output or unintended commission of input and are usually caused by internal failure of the components or failures propagated to the inputs of the components. For example, in Table 5, omission of standby recovery block is defined as:

$$\begin{aligned} \texttt{Omission-Output} = {} & \texttt{Omission-Primary.Output} \\ & \wedge \\ & \texttt{Omission-Standby.Output} \\ & \vee \\ & \texttt{Electro Magnetic Interference} \end{aligned}$$

Once the system model is annotated with failure behaviour, the second phase creates fault trees by examining the system model and the failure behaviour of the components. Output deviations from one component are connected to input deviations in others and thus HiP-HOPS can synthesise fault trees for the whole system from its constituent local failure annotations. The final phase takes the fault trees generated in the previous phase and analyses them to produce important information regarding the dependability of the system.

Now, for dynamic analysis, we need to create a dynamic model of the system with the help of the static model of the system which was already created for HiP-HOPS. The process of creating the dynamic model, i.e., behavioural mode chart, is shown in Figure 55. We can complete the whole process of dynamic analysis in three distinct phases: an annotation phase, a synthesis phase and an analysis phase. In the annotation phase, this thesis uses HiP-HOPS to model the functional mode chart of the system by identifying the potential failure scenarios that can violate the nominal behaviour of the system. The idea behind this is to identify *functional* and *degraded* modes, and thus compile them to create the mode chart. In the synthesis phase, the mode chart will be used as input to determine the potential causes of system failure from different modes (*functional or*

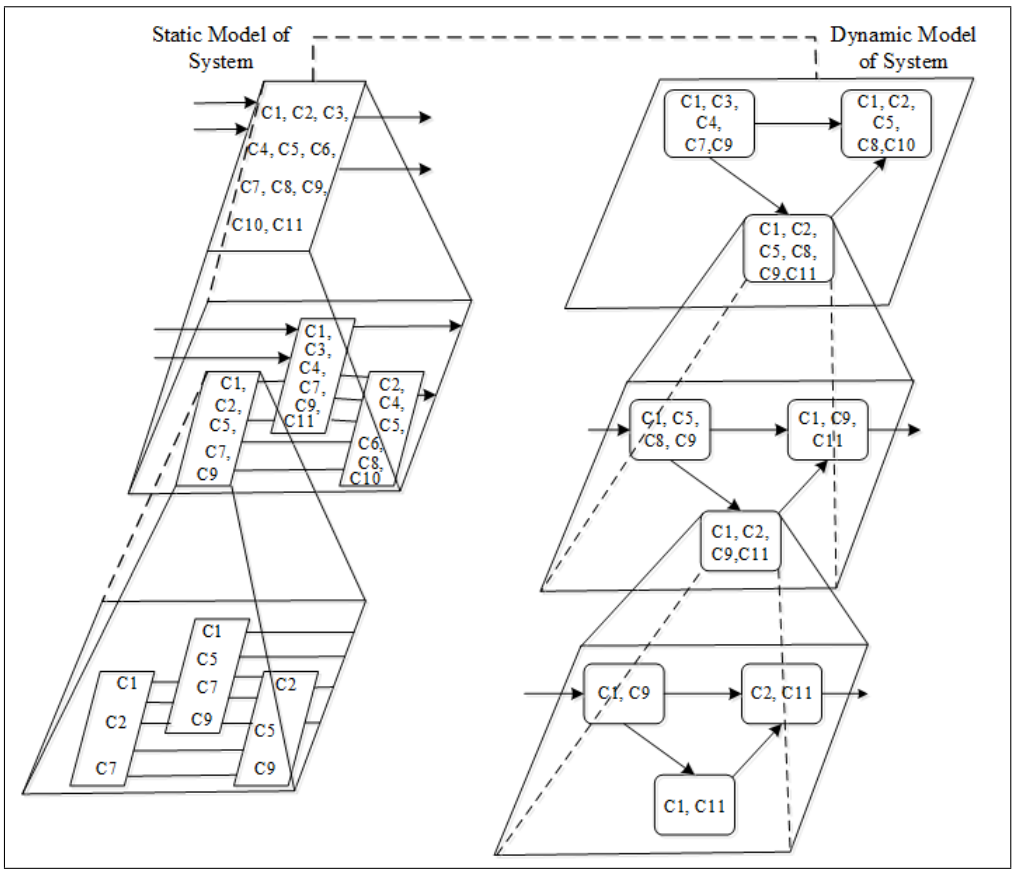


Figure 55: Mapping from static architectural model to dynamic mode chart

degraded). As the failure behaviour is determined based on modes and the modes are changed dynamically, at the end of this step we will obtain the dynamic analysis results of the system. Finally, the results (the fault trees) are analysed in the analysis phase to produce meaningful results about the dependability of the system.

3.4.1 The Annotation Phase

The HiP-HOPS static analysis provides the means to obtain a static model of the system and its failure behaviour. In our approach we will reuse the information obtained by the static analysis of the system. So, now we have the system architecture and the static behavioural annotation of the components in the architecture. We can see an example architecture in Figure 55 (static model) where the system has total 11 components (C1-C11) in the top level of the hierarchy. The system is decomposed into subsystems, and we can see that as we go down the system hierarchy, components are distributed among the subsystems. At a given point in time, the system may use one or more of its subsystems to provide a set of same or different functionality. That means that although the architecture is static, the functional configuration is dynamic. For example, consider the components at the lowest level of the hierarchy. At time *t*, the system may use a set of components {C1, C5, C9} to perform the task A, and at time *t+1* the system may use another set of components {C2, C7, C9} to perform the same task. That means the system architecture

remains static, but the configuration changes dynamically.

These distinct configurations represent different modes that the system could be in during its operation. Each of the modes consists of the name of the components involved (active) in that mode. Therefore, at a certain point in time, a mode does not necessarily represent the behaviour of a single component but rather it represents the behaviour of a group of interacting components, i.e., the behaviour of a distinct system configuration. As the modes represent different configurations of the system and the failure of a single or a group of components in a certain configuration can force the system to form a different configuration, failure of components will cause a mode transition. As the transitions are caused by the failure of the components, the transition conditions are derived from the failure behaviour annotation (done in the static analysis using HiP-HOPS) of the components.

The modelling process starts from the top of the static hierarchy and moves downwards. The parent modes or composite modes are identified from the top level of the static hierarchy. So, from the top level of the hierarchy we obtain the abstract mode chart of the system with all the component names in the respective modes. For small systems, a single mode chart is obtained for the whole architecture. For a relatively large system, as we go downward in the static hierarchy, the system is decomposed into smaller subsystems. Therefore, the configuration represented by different modes in the abstract mode chart will be divided into smaller configurations, resulting in sub-modes in the mode chart. Now, each of the sub-modes will represent a smaller configuration with a smaller number of components than their parent modes. For a particular parent mode, the sub-modes represent distinct system configurations and all these sub-modes (configurations) will act together to represent the configuration represented by the parent mode. This hierarchical decomposition will continue till no further decomposition is possible, i.e., no smaller configuration of subsystems is possible.

As mentioned earlier, all the components of a system are annotated with their failure behaviour for the static analysis using the HiP-HOPS. However, the components are now assigned to different modes in the mode chart and therefore their behaviour may need to be redefined. We refer the annotation done in the static analysis as the *default* behaviour of the components. If a component has different behaviour in different modes, then we have to define the mode based behaviour of that component. If no mode based behaviour is defined for a component, then the *default* behaviour of that component will be taken into account during the analysis. Let us consider the twin engine aircraft fuel system of Figure 41 again. The architecture of the system is formally defined in Figure 56.

Annotation of the components of the above architecture for static analysis is shown in Table 11. Output deviations and their logical expression are shown in Table 12. Now using HiP-HOPS, we can see that the system can continue to provide its functionality in three possible configurations. In the first configuration, the set of active components is {TL, TR, VL, VR, PL, PR}. As it represents a *functional* mode, we can refer it as **TLTR**

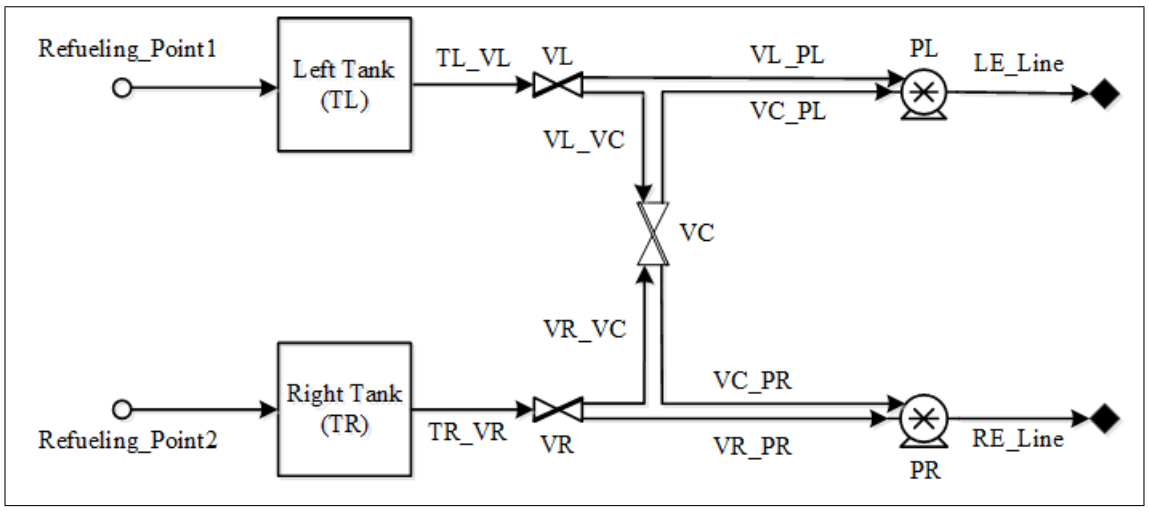


Figure 56: Architecture of twin engine aircraft fuel distribution system

Table 11: Failure behaviour of components of twin engine aircraft fuel distribution system

Component	Failure Behaviour	Failure Expression	Failure Rate
`Left Tank (TL)`	Internal failure or shortage of fuel	`TL.Empty` $\vee$ `TL.Blocked`	$\lambda_{TLE} \vee \lambda_{TLB}$
`Right Tank (TR)`	Internal failure or shortage of fuel	`TR.Empty` $\vee$ `TR.Blocked`	$\lambda_{TRE} \vee \lambda_{TRB}$
`VL`	Internal failure	`VL.Failed`	λ_{VL}
`VR`	Internal failure	`VR.Failed`	λ_{VR}
`VC`	Internal failure	`VC.Failed`	λ_{VC}
`PL`	Internal failure	`PL.Failed`	λ_{PL}
`PR`	Internal failure	`PR.Failed`	λ_{PR}

mode as both left tank (TL) and right tank (TR) are in use. The second configuration consists of the components set {TL, VL, VC, PL, PR}, and it is referred to as the mode **`TL`**. The final functional configuration of the system consists of the components set {TR, VR, VC, PL, PR} and corresponds to mode **`TR`**. As the failure modes of the components (see Table 11) are the same in any mode of operations, we do not need to refine them any further.

However, as the number of components varies from one mode to another and the system configuration changes, we have to redefine the output deviations of the system to show the mode based behaviour. In different configurations the system components may have to perform different roles, e.g., workload on a component may change from one mode to another. Consider, for instance, that each of the engines of the duel engine aircrafts

Table 12: Failure data for twin engine aircraft fuel distribution system

Output Deviations	Failure Expression
Omission-LE_Line	PL.Failed ∨ Omission-VL_PL ∧ Omission-VC_PL
Omission-RE_Line	PR.Failed ∨ Omission-VR_PR ∧ Omission-VC_PR
Omission-VL_PL	VL.Failed ∨ Omission-TL_VL
Omission-VR_PR	VR.Failed ∨ Omission-TR_VR
Omission-VC_PL	VC.Failed ∨ Omission-VR_VC
Omission-TL_VL	TL.Empty ∨ TL.Blocked
Omission-VR_VC	VR.Failed ∨ Omission-TR_VR
Omission-TR_VR	TR.Empty ∨ TR.Blocked
Omission-VL_VC	VL.Failed ∨ Omission-TL_VL
Omission-VC_PR	VC.Failed ∨ Omission-VL_VC

consumes *X* litres of fuel per minute. Therefore, if the system operates in **TLTR** mode, then *X* litres of fuel flows through both VL and VR. Now, if for some reason VR gets stuck closed then the system will switch to **TL** mode, which means the left tank will provide fuel to both the engines. This results in double (*2X* litres/minute) fuel flow through the VL, which means the pressure on valve VL is different on two different modes. At the same time, workload on some components may remain the same in different modes, i.e., their behaviour is mode-independent. For example, we may assume the workload on pump PL or PR remains the same in all the functional modes. For this reason we also have to redefine the failure rates of components to have mode based failure rate. As a result, a component can have the same failure logic expression in different modes but due to having different failure rates they are treated differently.

Following the instructions provided in subsection 3.3.2, the annotation of mode based failure behaviour of output deviations is performed and shown in Table 13. The redefined failure rates of the components are shown in Table 14.

Once all the *functional* modes and the mode based failure behaviour of the system are defined, then the transition conditions among the modes must be defined. For this particular example, we have three *functional* modes: **TLTR**, **TL**, and **TR**, where **TLTR** is the initial mode. Now we have to define how the **TL** and **TR** modes can be reached from the **TLTR** mode. Applying HiP-HOPS on the architecture of Figure 56, it is realised that the unavailability (failure) of line VR_PR will cause the system to go to **TL** mode from **TLTR** mode, and on the other hand, the unavailability of the line VL_PL will cause the system to go to the **TR** mode.

Table 13: Mode based expressions of failure behaviour

Failure Expression / **Output Deviations**	**TLTR**	**TL**	**TR**
Omission-LE_Line	PL.Failed	PL.Failed ∨ Omission-VL_PL	PL.Failed ∨ Omission-VC_PL
Omission-RE_Line	PR.Failed	PR.Failed ∨ Omission-VC_PR	PR.Failed ∨ Omission-VR_PR
Omission-VL_PL	VL.Failed ∨ Omission-TL_VL	VL.Failed ∨ Omission-TL_VL	No effect
Omission-VR_PR	VR.Failed ∨ Omission-TR_VR	No effect	VR.Failed ∨ Omission-TR_VR
Omission-VC_PL	No effect	No effect	VC.Failed ∨ Omission-VR_VC
Omission-TL_VL	TLeft.Empty ∨ TLeft.Blocked	TLeft.Empty ∨ TLeft.Blocked	No effect

Omission-VR_VC	No effect	No effect	VR.Failed ∨ Omission-TR_VR
Omission-TR_VR	TRight.Empty ∨ TRight.Blocked	No effect	TRight.Empty ∨ TRight.Blocked
Omission-VL_VC	No effect	VL.Failed ∨ Omission-TL_VL	No effect
Omission-VC_PR	No effect	VC. Failed ∨ Omission-VL_VC	No effect

Table 14: Mode based failure behaviour of components of twin engine aircraft fuel distribution system

Component	Failure Expression	Failure Rates		
		TLTR	TL	TR
Left Tank (TL)	TL.Empty ∨ TL.Blocked	$\lambda_{TLE} \vee \lambda_{TLB}$	$\lambda_{TLE1} \vee \lambda_{TLB1}$	-
Right Tank (TR)	TR.Empty ∨ TR.Blocked	$\lambda_{TRE} \vee \lambda_{TRB}$	-	$\lambda_{TRE1} \vee \lambda_{TRB1}$
VL	VL.Failed	λ_{VL1}	λ_{VL2}	-
VR	VR.Failed	λ_{VR1}	-	λ_{VR2}
VC	VC.Failed	-	λ_{VC}	λ_{VC}
PL	PL.Failed	λ_{PL}	λ_{PL}	λ_{PL}
PR	PR.Failed	λ_{PR}	λ_{PR}	λ_{PR}

If we denote the transitions from **TLTR** mode to **TL** and **TR** modes as T1 and T2 respectively, then their expression can be written using the predefined failure behaviour of the system as:

T1 = Omission-VR_PR

and

T2 = Omission-VL_PL

In the analysis, the mode chart is represented in a tabular form and the mode chart for the above example is shown in Table 15.

Table 15: Tabular representation of mode chart

Source Mode	Transition Condition	Destination Mode
TLTR	Omission-VR_PR	TL
TLTR	Omission-VL_PL	TR

3.4.2 The Synthesis Phase

Once the mode chart of the system architecture and the mode based failure data have been defined, the mode chart can then be synthesised using HiP-HOPS. This phase operates by examining how the failure of components propagates through different modes in the mode chart and causes system failure. Therefore, the first task of this phase is to identify the parts of the system model that act as the system outputs, since failures of these parts will be used as the top level failures (top events of the fault trees) of the whole system. In the case of the twin engine aircraft fuel distribution system, the supply of fuel to the

left engine can be considered as one of the system outputs. A failure to provide this functionality would be a failure of the system and so it would be treated as a starting point for the synthesis.

In the static analysis, for each system output, HiP-HOPS generates a local fault tree describing the reason for the output deviation. The tool then traverses the architecture backwards to examine the input components leading to the system output, and from there to the inputs of those components, and so forth. In this way the tool traverses the whole architecture and creates local fault trees for the interconnected components. The traversal continues until no connected components remains. After that, the tool goes back and combines the local fault trees into a single fault tree which represents all the possible combinations of component failure that can lead to the system failure. For dynamic analysis, the synthesis phase also creates a single fault tree showing the causes of each system failure, but the process of creating the fault trees differs from the static analysis.

In dynamic analysis, a given top event (system output) can potentially be caused by failures in multiple different modes. Just as HiP-HOPS traverses back through components in static synthesis, in dynamic synthesis HiP-HOPS will traverse back through modes as well, following each possible transition that leads to the failed mode representing the system failure. The result is a disjunction of fault trees that represents all the possible transitions from the starting state to the system failure mode. The top event of a mode specific fault tree is represented in the following form:

`Output Deviation Name<mode name>`

where the `mode name` defines the mode from which the causes for output deviation defined by the `Output Deviation Name` are required to be derived. For this example, if supply of fuel to the left engine is taken as a system output, then the set of fault trees representing this output deviation can be represented as:

`Omission-LE_Line ={Omission-LE_Line<TLTR>, Omission-LE_Line <TL>, Omission-LE_Line<TR>}`

In the above expression, `Omission-LE_Line` represents the output deviation of the system (omission of fuel in the line to the left engine) and each element of the set represents the cause of that output deviation in a particular mode. For example, `Omission-LE_Line<TLTR>` represents the cause for `Omission-LE_Line` from **`TLTR`** mode. As mentioned earlier, the elements of the set are actually the top events of their own fault trees, therefore, the whole tree for that top event is required to be developed. For this reason, the elements of the set are taken one by one and the fault trees are developed for each of the elements by traversing the system mode chart and system architecture. However, in this case, the fault tree development process varies from one element to another. If an element represents the top event of the fault tree for the *initial* mode of the mode chart, then the fault tree is generated by traversing the system architecture as

it is done in static analysis. That means the process starts with the failure logic of the system output that is defined for the initial mode. It then traverses the static architecture to examine the propagation of failure through the components to the system output. After that, the mode based local fault trees of all involved components are created and this process continues until no connected components remain. Once all the mode based local fault trees are created, then they are combined together to obtain a single fault tree.

For the duel engine aircraft fuel distribution system example, the failure expression of the `Omission-LE_Line` in the initial mode (**`TLTR`**) is defined as (see Table 13):

```
Omission-LE_Line<TLTR> = PL.Failed<TLTR>
```

That means the omission of system output from **`TLTR`** mode is caused by the internal failure of the left pump. Therefore, it is a single point of failure and we do not need to traverse the architecture any further.

Now, if we want to generate a fault tree for any internal modes (any modes other than the initial mode) in the mode chart, then we have to traverse both static system architecture and the mode chart itself (see Figure 57). This is because in this case, for a particular internal mode, we not only have to find the causes of system failure from that particular mode but also the causes for the system being in that particular mode. For example, in Figure 57, consider that we have to find the causes (in the form of a fault tree) that led the system to the system failure from Mode *i*. To do so, we have to find two things.

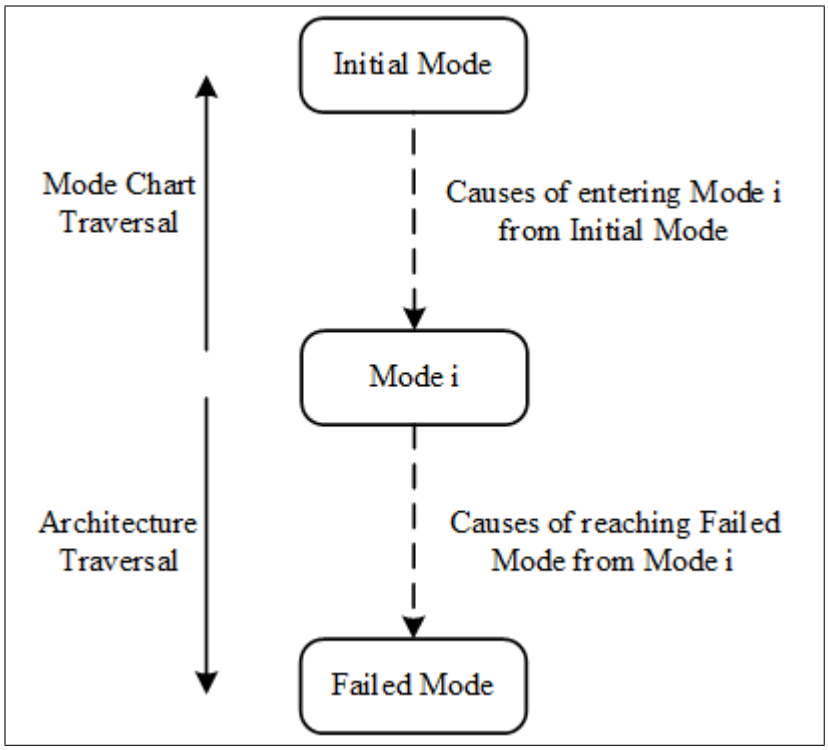


Figure 57: Fault tree generation process for any internal mode

Firstly, we need to find the causes of the system failure from Mode *i*, and secondly, we need to find the causes of the system being in Mode *i* from the initial mode. The former

is obtained by traversing the static architecture and the latter is obtained by traversing the system mode chart (see Figure 57). That means to obtain the fault tree showing the causes of system failure from a particular internal mode, we have to start with the failure logic expression of the output deviation of the system for that particular mode. Then the process traverses the static architecture and the mode chart at the same time. In this thesis, we refer these traversal processes as *architecture traversal* and *mode chart traversal*, respectively. In the *architecture traversal* step, it traverses the static architecture to examine the components that give rise to the input deviations received by the system output, and then creates mode based local fault trees for all the involved components. Once all the components are explored and the local fault trees are generated, then they are combined together to obtain a single fault tree. This concludes the first half of the task.

The remaining half of the task is done in the *mode chart traversal* step. In this step, the process starts with the mode for which we are obtaining the causes of system failure. Starting from this mode the process traverses the mode chart backward (towards the initial mode) to find the causes of the system being in this mode. In every iteration, the process replaces the current mode by its immediate preceding mode(s) and transition condition from the preceding mode(s) to the current mode. This process will continue until the initial mode is reached, and as a result of this process, we obtain all the possible combinations of events (component failures) that cause the system to go to the mode in question from the initial mode.

Consider the example of duel engine aircraft fuel distribution system. If we want to obtain the causes of system failure (`Omission-LE_Line`) from the **`TL`** mode then will start with the following expression:

$$\texttt{Omission-LE\_Line<TL>} = \texttt{TL} \triangleleft \texttt{Omission-LE\_Line<TL>}$$

On the right hand side of the above expression, the term inside the angle brackets means that the system has to be in the **`TL`** mode before we can obtain the failure behaviour of the system in **`TL`** mode. To ensure this sequence a PAND operator ($\triangleleft$) is used in the expression. The term following the PAND operator is the expression that causes the system failure from the **`TL`** mode. Now the process will proceed with the above expression in the following ways as described earlier.

Architecture Traversal Step:

$$\texttt{Omission-LE\_Line<TL>} = \texttt{PL.Failed<TL>} \vee \texttt{Omission-VL\_PL<TL>}$$
$$= \texttt{PL.Failed<TL>} \vee \texttt{VL.Failed<TL>} \vee \texttt{Omission-TL\_VL<TL>}$$
$$= \texttt{PL.Failed<TL>} \vee \texttt{VL.Failed<TL>} \vee \texttt{TLeft.Empty<TL>} \vee \texttt{TLeft.Blocked<TL>}$$

Mode Chart Traversal Step:

$$\texttt{\{TL\}} = \texttt{\{TLTR\}.T1}$$
$$= \texttt{Omission-VR\_PR<TLTR>}$$

= VR.Failed<TLTR> ∨ Omission-TR_VR<TLTR>)
= VR.Failed<TLTR> ∨ TRight.Empty<TLTR> ∨
TRight.Blocked<TLTR>

Now combining the results obtained by the architecture traversal and the mode chart traversal steps the failure behaviour of the system output from the **TL** mode is written as:

Omission-LE_Line<TL>= (VR.Failed<TLTR> ∨ TRight.Empty<TLTR>
∨ TRight.Blocked<TLTR>) ◁ (PL.Failed<TL> ∨ VL.Failed<TL>
∨ TLeft.Empty<TL> ∨ TLeft.Blocked<TL>)

In the same way the causes of system failure from the **TR** mode can be obtained as:

Omission-LE_Line<TR>= (VL.Failed<TLTR> ∨ TLeft.Empty<TLTR>
∨ TLeft.Blocked<TLTR>) ◁ (PL.Failed<TR> ∨ VC.Failed<TR>
∨ VR.Failed<TR> ∨ TRight.Empty<TR> ∨ TRight.Blocked<TR>)

Now the complete failure behaviour of the system output can be obtained by taking logical OR of the individual failure behaviour in different modes.

Omission-LE_Line = Omission-LE_Line<TLTR> ∨
Omission-LE_Line<TL> ∨ Omission-LE_Line<TR>
= PL.Failed<TLTR> ∨ (VR.Failed<TLTR> ∨ TRight.Empty<TLTR>
∨ TRight.Blocked<TLTR>) ◁ (PL.Failed<TL> ∨ VL.Failed<TL> ∨
TLeft.Empty<TL> ∨ TLeft.Blocked<TL>) ∨ (VL.Failed<TLTR>
∨ TLeft.Empty<TLTR> ∨ TLeft.Blocked<TLTR>) ◁ (PL.Failed<TR>
∨ VC.Failed<TR> ∨ VR.Failed<TR> ∨ TRight.Empty<TR>
∨ TRight.Blocked<TR>)

From Table 14, it is seen that the failure rate of PL is same in all the modes, i.e., behaviour of PL is mode-independent. In that case, PL.Failed can be used to represent the same failure in any mode, thus PL.Failed ⟺ PL.Failed<TLTR> ⟺ PL.Failed<TR> ⟺ PL.Failed<TL>. Therefore, the above expression can be rewritten as:

= PL.Failed ∨ (VR.Failed<TLTR> ∨ TRight.Empty<TLTR>
∨ TRight.Blocked<TLTR>) ◁ (PL.Failed ∨ VL.Failed<TL> ∨
TLeft.Empty<TL> ∨ TLeft.Blocked<TL>) ∨ (VL.Failed<TLTR>
∨ TLeft.Empty<TLTR> ∨ TLeft.Blocked<TLTR>) ◁ (PL.Failed

∨ VC.Failed<TR> ∨ VR.Failed<TR> ∨ TRight.Empty<TR>
∨ TRight.Blocked<TR>)

This fault tree expression now shows the causes of the omission of fuel to the left engine from all relevant modes.

3.4.3 The Analysis Phase

The outcome of the synthesis process is a set of one or more interconnected Pandora temporal fault trees (TFTs). Qualitative analysis of classical fault trees allows the analysts to draw conclusions about the possible causes of system failure in the form of combinations of events (component failure). As a fault tree can contain many branches and many gates, qualitative analysis simplifies and transforms the fault tree into a set of minimal cut sets. Each minimal cut set is the smallest combinations of events that can cause the system failure and is represented as the conjunction of a number of events. As HiP-HOPS is a fault tree analysis tool, it also analyses the fault trees to obtain minimal cut sets and it uses a variation of MICSUP (Pande *et al.*, 1975) as its primary minimal cut set generation algorithm.

For the dynamic analysis, as the synthesis process produces Pandora temporal fault trees instead of classical fault trees, the existing FTA techniques used in HiP-HOPS cannot be used to analyse the temporal fault trees. This is because of the event ordering and the presence of the temporal gates in the Pandora TFTs. In Pandora, the term cut sequence (CSQ) replaces the term cut set (CS) of classical fault trees. Now, CSQ represents a sequence of events that can cause the system failure. The smallest sequence of events that is necessary and sufficient to cause the top event is termed the minimal cut sequence (MCSQ). Therefore, we now need to perform qualitative analysis of Pandora TFTs and the goal is to obtain minimal cut sequences. The ways of performing qualitative analysis of Pandora TFTs have been shown by Walker (2009, Chapter 4). In the analysis phase, we use the temporal fault trees generated in the synthesis phase and the methodologies proposed by Walker (2009) to obtain the minimal cut sequences.

Using a prototype version of the HiP-HOPS tool the MCSQs to cause the top event `Omission-LE_Line` are calculated to be:

1. PL.Failed
2. VR.Failed<TLTR> ◁ VL.Failed<TL> ∧ VR.Failed<TLTR>
   ≀ TLeft.Empty<TL> ∧ VR.Failed<TLTR> ≀ TLeft.Blocked<TL>
3. VR.Failed<TLTR> ≀ VL.Failed<TL> ∧ VR.Failed<TLTR>
   ◁ TLeft.Empty<TL> ∧ VR.Failed<TLTR> ≀ TLeft.Blocked<TL>
4. VR.Failed<TLTR> ≀ VL.Failed<TL> ∧ VR.Failed<TLTR>
   ≀ TLeft.Empty<TL> ∧ VR.Failed<TLTR> ◁ TLeft.Blocked<TL>
5. TRight.Empty<TLTR> ◁ VL.Failed<TL> ∧ TRight.Empty<TLTR>

≀ TLeft.Empty<TL> ∧ TRight.Empty<TLTR> ≀ TLeft.Blocked<TL>

6. TRight.Empty<TLTR> ≀ VL.Failed<TL> ∧ TRight.Empty<TLTR>
   ◁ TLeft.Empty<TL> ∧ TRight.Empty<TLTR> ≀ TLeft.Blocked<TL>
7. TRight.Empty<TLTR> ≀ VL.Failed<TL> ∧ TRight.Empty<TLTR>
   ≀ TLeft.Empty<TL> ∧ TRight.Empty<TLTR> ◁ TLeft.Blocked<TL>
8. TRight.Blocked<TLTR> ◁ VL.Failed<TL> ∧ TRight.Blocked<TLTR>
   ≀ TLeft.Empty<TL> ∧ TRight.Blocked<TLTR> ≀ TLeft.Blocked<TL>
9. TRight.Blocked<TLTR> ≀ VL.Failed<TL> ∧ TRight.Blocked<TLTR>
   ◁ TLeft.Empty<TL> ∧ TRight.Blocked<TLTR> ≀ TLeft.Blocked<TL>
10. TRight.Blocked<TLTR> ≀ VL.Failed<TL> ∧ TRight.Blocked<TLTR>
    ≀ TLeft.Empty<TL> ∧ TRight.Blocked<TLTR> ◁ TLeft.Blocked<TL>
11. VL.Failed<TLTR> ◁ VC.Failed<TR> ∧ VL.Failed<TLTR>
    ≀ VR.Failed<TR> ∧ VL.Failed<TLTR> ≀ TRight.Empty<TR>
    ∧ VL.Failed<TLTR> ≀ TRight.Blocked<TR>
12. VL.Failed<TLTR> ≀ VC.Failed<TR> ∧ VL.Failed<TLTR>
    ◁ VR.Failed<TR> ∧ VL.Failed<TLTR> ≀ TRight.Empty<TR>
    ∧ VL.Failed<TLTR> ≀ TRight.Blocked<TR>
13. VL.Failed<TLTR> ≀ VC.Failed<TR> ∧ VL.Failed<TLTR>
    ≀ VR.Failed<TR> ∧ VL.Failed<TLTR> ◁ TRight.Empty<TR>
    ∧ VL.Failed<TLTR> ≀ TRight.Blocked<TR>
14. VL.Failed<TLTR> ≀ VC.Failed<TR> ∧ VL.Failed<TLTR>
    ≀ VR.Failed<TR> ∧ VL.Failed<TLTR> ≀ TRight.Empty<TR>
    ∧ VL.Failed<TLTR> ◁ TRight.Blocked<TR>
15. TLeft.Empty<TLTR> ◁ VC.Failed<TR> ∧ TLeft.Empty<TLTR>
    ≀ VR.Failed<TR> ∧ TLeft.Empty<TLTR> ≀ TRight.Empty<TR>
    ∧ TLeft.Empty<TLTR> ≀ TRight.Blocked<TR>
16. TLeft.Empty<TLTR> ≀ VC.Failed<TR> ∧ TLeft.Empty<TLTR>
    ◁ VR.Failed<TR> ∧ TLeft.Empty<TLTR> ≀ TRight.Empty<TR>
    ∧ TLeft.Empty<TLTR> ≀ TRight.Blocked<TR>
17. TLeft.Empty<TLTR> ≀ VC.Failed<TR> ∧ TLeft.Empty<TLTR>
    ≀ VR.Failed<TR> ∧ TLeft.Empty<TLTR> ◁ TRight.Empty<TR>
    ∧ TLeft.Empty<TLTR> ≀ TRight.Blocked<TR>
18. TLeft.Empty<TLTR> ≀ VC.Failed<TR> ∧ TLeft.Empty<TLTR>
    ≀ VR.Failed<TR> ∧ TLeft.Empty<TLTR> ≀ TRight.Empty<TR>
    ∧ TLeft.Empty<TLTR> ◁ TRight.Blocked<TR>
19. TLeft.Blocked<TLTR> ◁ VC.Failed<TR> ∧ TLeft.Blocked<TLTR>
    ≀ VR.Failed<TR> ∧ TLeft.Blocked<TLTR> ≀ TRight.Empty<TR>
    ∧ TLeft.Blocked<TLTR> ≀ TRight.Blocked<TR>
20. TLeft.Blocked<TLTR> ≀ VC.Failed<TR> ∧ TLeft.Blocked<TLTR>
    ◁ VR.Failed<TR> ∧ TLeft.Blocked<TLTR> ≀ TRight.Empty<TR>

∧ TLeft.Blocked<TLTR> ≀ TRight.Blocked<TR>

21. TLeft.Blocked<TLTR> ≀ VC.Failed<TR> ∧ TLeft.Blocked<TLTR> ≀ VR.Failed<TR> ∧ TLeft.Blocked<TLTR> ◁ TRight.Empty<TR> ∧ TLeft.Blocked<TLTR> ≀ TRight.Blocked<TR>
22. TLeft.Blocked<TLTR> ≀ VC.Failed<TR> ∧ TLeft.Blocked<TLTR> ≀ VR.Failed<TR> ∧ TLeft.Blocked<TLTR> ≀ TRight.Empty<TR> ∧ TLeft.Blocked<TLTR> ◁ TRight.Blocked<TR>

3.5 Case Study

To illustrate the use of HiP-HOPS to perform dynamic modelling and analysis of a more complex system, we consider an Aircraft Fuel Distribution System (AFDS), first presented in (Papadopoulos, 2000) but reworked here, and shown in Figure 58. As seen in the figure, the system model represents the fuel storage and distribution system of a twin engine aircraft. The system consists of:

1. Seven identical (in capacity) fuel tanks to store fuel and they are connected with polythene piping. To maintain a balance across the body of the aircraft, the fuel tanks are distributed symmetrically across the vertical and the horizontal axes. As seen in the figure, in the horizontal axis, there are two tanks in each of the wings of the aircraft and in the vertical axis there are tanks in the front, central, and rear parts of the aircraft. Each tank is coupled with a level sensor to measure the level of the fuel in the tank.

2. Eleven bi-directional fuel pumps embedded with speed sensor to provide fuel flow throughout the system, especially towards the engines. The flow rates and the direction of flows are computer controlled to meet the demand of the engines and also to maintain the level of fuel in all the tanks to avoid imbalance in the body of the aircraft.

3. Twelve valves that can be used to activate some paths or block some paths according to the requirement of the system in different situations. They are software controlled and their operations are synchronised with the operations of pumps to ensure effective collaboration.

4. Twelve flow meters to measure the rate of fuel flow through the pipes and these measurements are used in deciding the paths to activate and deactivate to maintain the proper fuel flow to the engines and maintaining balance across the body of the aircraft.

5. A refuelling point to refill the tanks.

6. Two jettison points to release fuel to the environment in case of any emergency conditions either to maintain the balance across the body of the aircraft or to reduce the weight of the aircraft to ease the emergency landing.

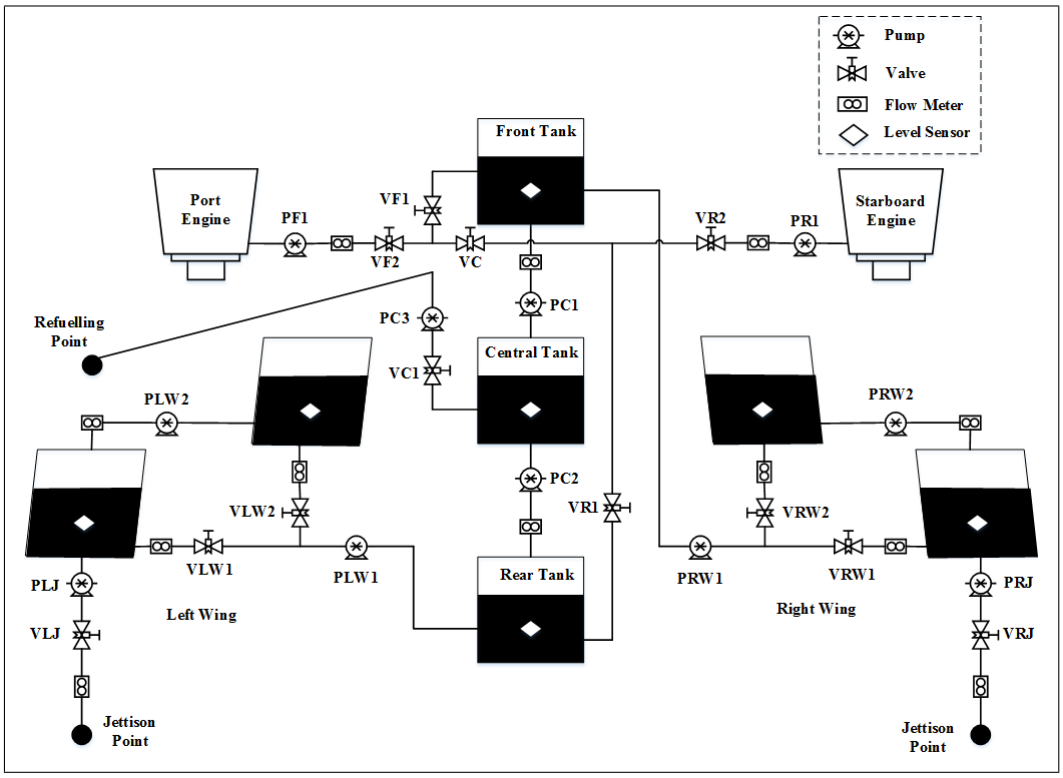


Figure 58: Aircraft Fuel Distribution System (Papadopoulos, 2000)

The AFDS under consideration has two main functions: storing fuel in the tanks and distributing the fuel throughout the system. This functionality is provided in two different modes — refuelling and consumption. The refuelling takes place when the aircraft is parked (pre-flight). It is done by injecting fuel into the central tank and then automatically distributing the fuel to the different tanks. The consumption of fuel takes place during a flight, which includes the taxiing, take-off, climbing, cruising, approaching and landing phases. Under normal operation, the front tank is feeding the Port Engine (PE) and the Starboard Engine (SE) is fed from the rear tank. At this time the central valve (VC) and the jettison valves (VLJ and VRJ) are closed. As fuel is consumed by the engines, the fuel level in both the front and rear tanks will go down. To maintain the appropriate fuel level in the front and rear tanks, fuel is transferred to these two tanks from the central tank and the tanks in the wings. Fuel to the front tank comes from the central tank and the right wing tanks, on the other hand the rear tank gets fuel from the central tank and the left wing tanks. In normal operating conditions, the distribution of fuel is done in a way that the centre of gravity remains near the centre of the body of the aircraft.

In conditions of failure, both the engines can be fed from a single tank. For example, if the valve VF1 fails stuck closed, then it is possible to open the valve VC to cross-feed both the engines from the rear tank. Although, in this case, the flow from the rear tank

is needed to be doubled to meet the fuel demand of two engines. As a result, the system becomes imbalanced and it is possible to direct the fuel flow from different sources to various destinations with variable speed to maintain the balance. In normal conditions, both symmetrically positioned (in outer wing tanks) jettison paths are deactivated by closing the jettison valves (VLJ and VRJ). However, in any emergency condition, the system allows the jettison of fuel to the atmosphere by opening the corresponding valves. Jettison of fuel usually takes place during an emergency landing when it is necessary to discard the extra fuel to reduce the mass of the aircraft to ease the safe landing. That means in the presence of failure, the system configuration is needed to be changed by activating and/or deactivating system components to compensate for the effects of the failure and to continue system operation either with equal or degraded functionality. The activation or deactivation or any other calibration of the system components is done by a central computerised control system. Therefore, this centralised control system represents a single point of failure. However, apart from this single point of failure, the failure behaviour of the system is more interesting in the more refined levels of detail. Different fault-tolerant strategies are implemented in the system architecture, hence, we can see that there are redundant components and redundant paths of fuel flow to tolerate some faults up to some certain levels. Therefore, in this section, we will concentrate on determining the mode based failure behaviour of the system at the plant level, i.e., in different operational modes, what are the possible combinations of failures that occur at plant level (e.g., failure of pump, valve malfunctions) that can cause the system failure.

3.5.1 The System Architecture and the Static Hierarchy

The static hierarchy of the fuel distribution system shown in Figures 59 and 60 were firstly shown in (Papadopoulos, 2000). In the first level of the hierarchy, the system is represented as a single block with one input and four outputs. The input comes from the refuelling point and represents the flow of fuel to the system in the refuelling phase. Two of the outputs shows the flow of fuel to the two engines and two remaining outputs show the fuel flow to the jettison points. In the second level of the hierarchy, the system is decomposed into four subsystems and now the input and the outputs are connected to different subsystems. Different subsystems are now doing different tasks. The engine feed cross-feed (EFCF) subsystem is directly involved in feeding the engines with fuel, and as seen in Figure 60, the two output lines — `Starboad_Engine_Line` and `Port_Engine_Line` from the EFCF subsystems represent two fuel flow path to starboard and port engine respectively.

The jettison lines are now going out from the Right Wing Deposit (RWD) and the Left Wing Deposit (LWD). The input line (`Refuelling_Line`) is directly connected to the Central Deposit (CD), that means in the refuelling phase, fuel is provided to central deposit and it then distributes the fuel through two output lines (`Central_Line_1` and `Central_Line_2`). Each subsystem consists of a variable number of components

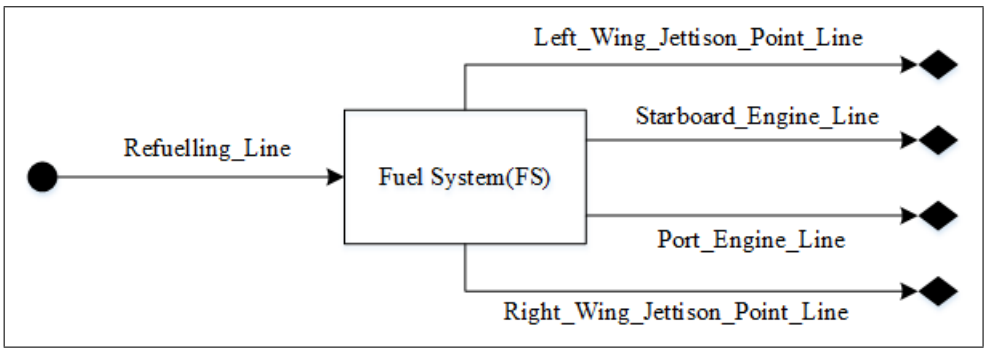


Figure 59: First level of the static hierarchy of the fuel distribution system (Papadopoulos, 2000)

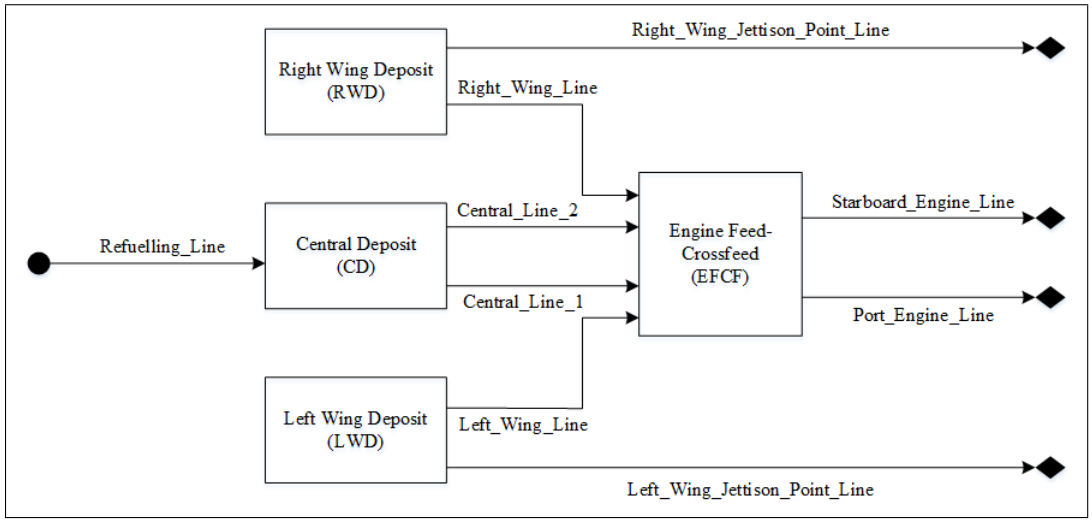


Figure 60: Second level of the static hierarchy of the fuel distribution system (Papadopoulos, 2000)

and is responsible for performing a particular set of operations. The component-level decomposition of the CD, EFCF, and RWD subsystem are shown in Figures 61, 62, and 63 respectively.

As the structure of the LWD subsystem is similar to that of the RWD subsystem, the component level decomposition is not shown here. In this thesis, we will analyse the mode based failure behaviour of the EFCF subsystem and therefore from now on we will concentrate only on this particular subsystem.

3.5.2 Static Analysis of the EFCF Subsystem

To perform the static analysis of the EFCF subsystem, we need to annotate the components of this subsystem with their failure information. After that, we have to identify the output of this subsystem. As mentioned earlier, the EFCF is responsible for providing fuel to the starboard and port engine. So, provision of fuel to each engine can be considered as the output of this subsystem, and thus failure of EFCF to provide fuel to any of the engines could be considered as a hazardous situation. As the fuel to the port engine and the starboard engine is provided in a similar fashion with the opposite set of components,

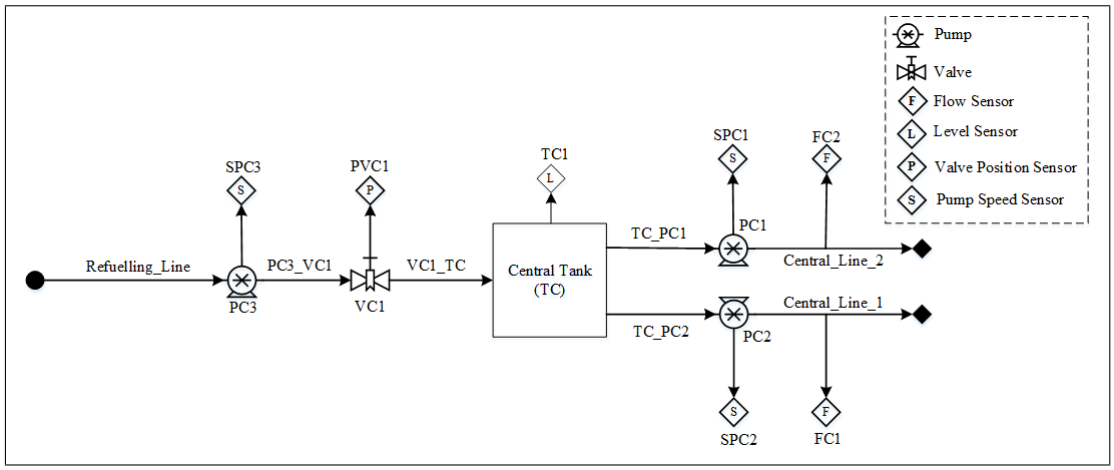


Figure 61: Component level decomposition of CD subsystem (Papadopoulos, 2000)

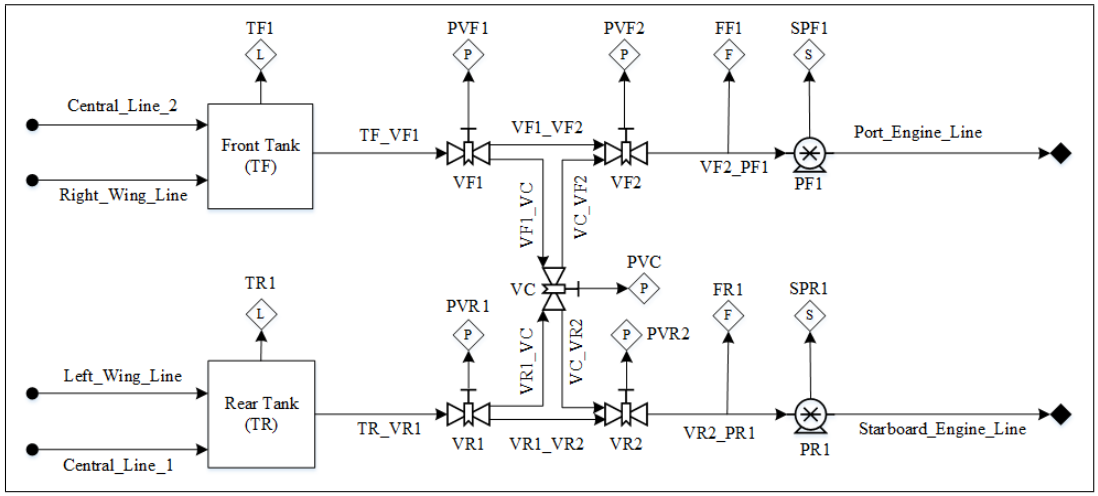


Figure 62: Component level decomposition of EFCF subsystem (Papadopoulos, 2000)

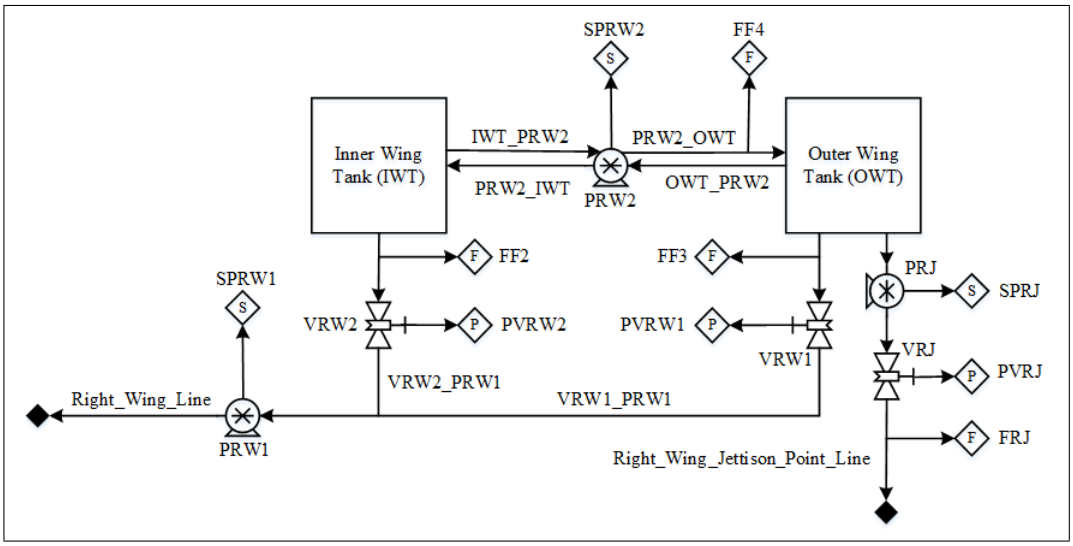


Figure 63: Component level decomposition of RWD subsystem (Papadopoulos, 2000)

we will concentrate on the failure of the EFCF subsystem to provide fuel to the port engine alone. The failure data for different output deviations are shown in Table 16 and the failure behaviour of the components of the EFCF subsystem is shown in Table 17.

Table 16: Failure data for output deviations of EFCF subsystem

Output Deviations	**Failure Expression**
Omission-Port_ Engine_Line	PF1.Failed ∨ Omission-VF2_PF1
Omission- Starboad_ Engine_Line	PR1.Failed ∨ Omission-VR2_PR1
Omission-VF2_PF1	VF2.Failed ∨ Omission-VF1_VF2 ∧ Omission-VC_VF2
Omission-VR2_PR1	VR2.Failed ∨ Omission-VR1_VR2
Omission-VF1_VF2	VF1.Failed ∨ Omission-TF_VF1
Omission-VR1_VR2	VR1.Failed ∨ Omission-TR_VR1
Omission-TF_VF1	TF.Failed
Omission-TR_VR1	TR.Failed
Omission-VF1_VC	VF1.Failed ∨ Omission-TF_VF1
Omission-VC_VR2	VC.Failed ∨ Omission-VF1_VC
Omission-VR1_VC	VR1.Failed ∨ Omission-TR_VR1
Omission-VC_VF2	VC.Failed ∨ Omission-VR1_VC

From the system architecture and using the data from tables 16 and 17, it is possible to mechanically derive a fault tree using HiP-HOPS for the condition “no fuel provided to port engine”, i.e., the combination of component failure that can cause Omission-Port _Engine Line. One thing to note is that in the analysis the flow sensors are considered as reliable, i.e., in any situation they will give a proper reading. Figure 64 shows the fault tree for the condition “no fuel to the port engine”.

Table 17: Failure behaviour of the components of the EFCF subsystem

Component	Failure Behaviour	Failure Expression	Failure Rate
Front Tank (TF)	Internal failure or shortage of fuel	TF.Empty ∨ TF.Blocked	$\lambda_{TFE} \vee \lambda_{TFB}$
Rear Tank (TR)	Internal failure or shortage of fuel	TR.Empty ∨ TR.Blocked	$\lambda_{TRE} \vee \lambda_{TRB}$
VF1	Internal failure or commission of input	VF1.Stuck_Closed ∨ VF1.Comission-close_command	$\lambda_{VF1SC} \vee \lambda_{VF1CC}$
VF2	Internal failure or commission of input	VF2.Stuck_Closed ∨ VF2.Comission-close_command	$\lambda_{VF2SC} \vee \lambda_{VF2CC}$
VC	Internal failure or commission of input	VC.Stuck_Closed ∨ VC.Comission-close_command	$\lambda_{VCSC} \vee \lambda_{VCCC}$
PF1	Internal failure	PF1.Electromechanical_fail_stop ∨ PF1.Control_valve_stuck_at_zero	$\lambda_{PF1EMF} \vee \lambda_{PF1@SZ}$
VR1	Internal failure or commission of input	VR1.Stuck_Closed ∨ VR1.Comission-close_command	$\lambda_{VR1SC} \vee \lambda_{VR1CC}$
VR2	Internal failure or commission of input	VR2.Stuck_Closed ∨ VR2.Comission-close_command	$\lambda_{VR2SC} \vee \lambda_{VR2CC}$
PR1	Internal failure	PR1.Electromechanical_fail_stop ∨ PR1.Control_valve_stuck_at_zero	$\lambda_{PR1EMF} \vee \lambda_{PR1@SZ}$

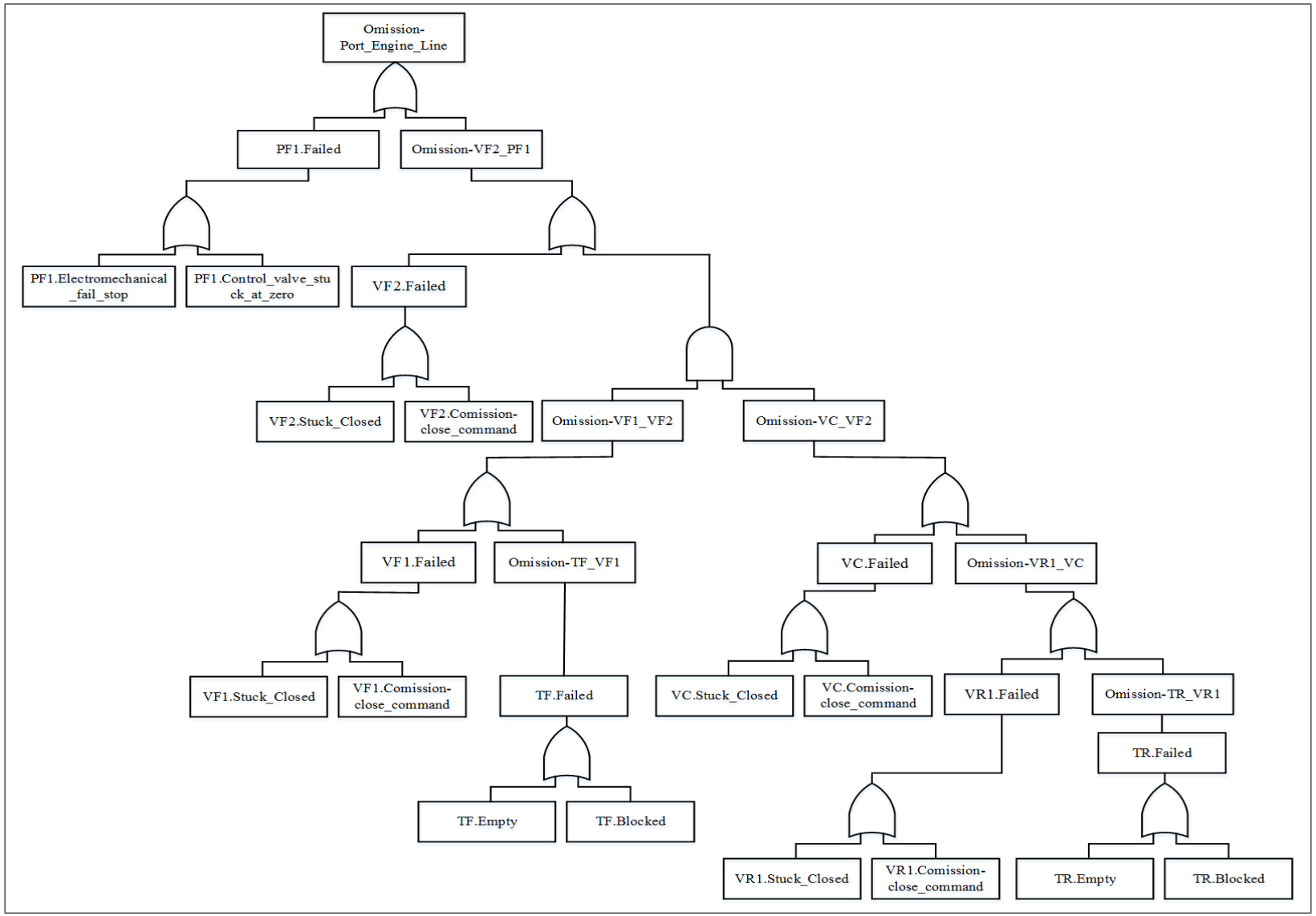


Figure 64: Fault tree of the condition “no fuel to the port engine”

The fault tree of Figure 64 can be analysed to obtain the minimal cut sets. The analysis obtains 28 cut sets as shown below. Each of these cut sets can cause the condition '*no fuel flow to the port engine*'. If we inspect the system architecture at Figure 62, then it is obvious that the failure of pump PF1 (`PF1.Failed`) or failure of flow through VF2 due to the failure of VF2 (`VF2.Failed`) will always cause no fuel flow to the port engine and these conditions are represented by cut sets 1-4. The rest of the cut sets represent the failure expression as conjunction of two events. For example, cut set 5 represents that the failure of valve VF1 (`VF1.Stuck_Closed`) and VC (`VC.Stuck_Closed`) can cause no fuel flow to the port engine.
Using the HiP-HOPS tool the Minimal Cut Sets to cause the top event (no fuel to the port engine) are calculated to be:

1. `PF1.Electromechanical_fail_stop`
2. `PF1.Control_valve_stuck_at_zero`
3. `VF2.Stuck_Closed`
4. `VF2.Comission-close_command`
5. `VF1.Stuck_Closed` ∧ `VC.Stuck_Closed`
6. `VF1.Stuck_Closed` ∧ `VC.Comission-close_command`
7. `VF1.Stuck_Closed` ∧ `VR1.Stuck_Closed`
8. `VF1.Stuck_Closed` ∧ `VR1.Comission-close_command`
9. `VF1.Stuck_Closed` ∧ `TR.Empty`
10. `VF1.Stuck_Closed` ∧ `TR.Blocked`
11. `VF1.Comission-close_command` ∧ `VC.Stuck_Closed`
12. `VF1.Comission-close_command` ∧ `VC.Comission-close_command`
13. `VF1.Comission-close_command` ∧ `VR1.Stuck_Closed`
14. `VF1.Comission-close_command` ∧ `VR1.Comission-close_command`
15. `VF1.Comission-close_command` ∧ `TR.Empty`
16. `VF1.Comission-close_command` ∧ `TR.Blocked`
17. `TF.Empty` ∧ `VC.Stuck_Closed`
18. `TF.Empty` ∧ `VC.Comission-close_command`
19. `TF.Empty` ∧ `VR1.Stuck_Closed`
20. `TF.Empty` ∧ `VR1.Comission-close_command`
21. `TF.Empty` ∧ `TR.Empty`
22. `TF.Empty` ∧ `TR.Blocked`
23. `TF.Blocked` ∧ `VC.Stuck_Closed`
24. `TF.Blocked` ∧ `VC.Comission-close_command`
25. `TF.Blocked` ∧ `VR1.Stuck_Closed`
26. `TF.Blocked` ∧ `VR1.Comission-close_command`
27. `TF.Blocked` ∧ `TR.Empty`
28. `TF.Blocked` ∧ `TR.Blocked`

3.5.3 Dynamic Analysis of the EFCF Subsystem

According to the subsection 3.4.1, in order to perform the dynamic analysis, we need to determine the different functional configurations of the system from the static analysis and the system architecture. At the same time, it is also necessary to obtain the causes of each change in configuration. Different functional configurations will represent the modes and the causes of configuration changes will act as transition conditions in the mode chart. From the description of the system, it is clear that in the normal operating condition the port engine is provided with fuel from the front tank and the starboard engine from the rear tank. However, in the condition of failure the EFCF subsystem can cross-feed both the engines. Therefore, broadly we can divide the operation of the EFCF subsystem into two modes — normal mode and cross-feed mode. In the normal mode, both the tanks (front and rear) are in use. However, in the cross-feed mode, there are two possibilities. The first possibility is that both the engines are fed from the front tank. This possibility occurs when the starboard engine does not receive any fuel from the rear tank. In the second possibility, both engines are fed from the rear tank because the port engine is not getting fuel from the front tank. So, we have three modes of operations for the EFCF subsystem, namely `EFCF_Normal`, `EFCF_Front_Tank_Cross_Feed`, and `EFCF_Rear_Tank_Cross_Feed`. Now we have to define the mode based behaviour of the components to facilitate dynamic analysis. The first step is to determine the set of components that are used in different modes to define their behaviour in the respective modes. The sets of components used in different modes are as below:

`EFCF_Normal = {TF, TR, VF1, VR1, VF2, VR2, PF1, PR1}`
`EFCF_Front_Tank_Cross_Feed = { TF, VF1, VC, VF2, VR2, PF1, PR1}`
`EFCF_Rear_Tank_Cross_Feed = {TR, VR1, VC, VF2, VR2, PF1, PR1}`

The mode based failure behaviour of the components are shown in Table 18. We can see that the failure logic of the components remains the same in different modes; however, their failure rates change depending on the configuration. Table 19 shows the mode based logical expression for different output deviations. Now we have to identify the transition conditions between the modes in the mode chart. The EFCF subsystem starts its operation in `EFCF_Normal (EFCF_N)` mode. Some output deviations of the subsystem would force it to go either to the `EFCF_Front_Tank_Cross_Feed(EFCF_FTCF)` or `EFCF_Rear_Tank_Cross_Feed(EFCF_RTCF)` mode. From static analysis it is identified that the EFCF subsystem would go to the `EFCF_Front_Tank_Cross_Feed` mode if `Omission-VR1_VR2` occurs, on the other hand, `Omission-VF1_VF2` would cause the system to go the `EFCF_Rear_Tank_Cross_Feed` mode. The functional mode chart of the EFCF subsystem is shown in Figure 65 and the tabular form of the mode chart is shown in Table 20.

Table 18: Mode based failure behaviour of components of EFCF subsystem

Component	Failure Expression	Failure Rates		
		EFCF_N	**EFCF_FTCF**	**EFCF_RTCF**
Front Tank (TF)	TF.Empty ∨ TF.Blocked	$\lambda_{TFE} \vee \lambda_{TFB}$	$\lambda_{TFE1} \vee \lambda_{TFB1}$	-
Rear Tank (TR)	TR.Empty ∨ TR.Blocked	$\lambda_{TRE} \vee \lambda_{TRB}$	-	$\lambda_{TRE1} \vee \lambda_{TRB1}$
VF1	VF1.Stuck_Closed ∨ VF1.Comission-close_command	$\lambda_{VF1SC} \vee \lambda_{VF1CC}$	$\lambda_{VF1SC1} \vee \lambda_{VF1CC1}$	-
VF2	VF2.Stuck_Closed ∨ VF2.Comission-close_command	$\lambda_{VF2SC} \vee \lambda_{VF2CC}$	$\lambda_{VF2SC2} \vee \lambda_{VF2CC2}$	$\lambda_{VF2SC3} \vee \lambda_{VF2CC3}$
VC	VC.Stuck_Closed ∨ VC.Comission-close_command	-	$\lambda_{VCSC1} \vee \lambda_{VCCC1}$	$\lambda_{VCSC2} \vee \lambda_{VCCC2}$
PF1	PF1.Electromechanical_fail_stop ∨ PF1.Control_valve_stuck_at_zero	$\lambda_{PF1EMF} \vee \lambda_{PF1S@Z}$	$\lambda_{PF1EMF} \vee \lambda_{PF1S@Z}$	$\lambda_{PF1EMF} \vee \lambda_{PF1S@Z}$
VR1	VR1.Stuck_Closed ∨ VR1.Comission-close_command	$\lambda_{VR1SC} \vee \lambda_{VR1CC}$	-	$\lambda_{VR1SC1} \vee \lambda_{VR1CC1}$
VR2	VR2.Stuck_Closed ∨ VR2.Comission-close_command	$\lambda_{VR2SC} \vee \lambda_{VR2CC}$	$\lambda_{VR2SC2} \vee \lambda_{VR2CC2}$	$\lambda_{VR2SC3} \vee \lambda_{VR2CC3}$
PR1	PR1.Electromechanical_fail_stop ∨ PR1.Control_valve_stuck_at_zero	$\lambda_{PR1EMF} \vee \lambda_{PR1S@Z}$	$\lambda_{PR1EMF} \vee \lambda_{PR1S@Z}$	$\lambda_{PR1EMF} \vee \lambda_{PR1S@Z}$

Table 19: Mode based failure data for the EFCF subsystem

Failure Expression / Output Deviations	**EFCF_N**	**EFCF_FTCF**	**EFCF_RTCF**
Omission-Port_Engine_Line	PF1.Failed ∨ Omission-VF2_PF1	PF1.Failed ∨ Omission-VF2_PF1	PF1.Failed ∨ Omission-VF2_PF1
Omission- Starboad_Engine_Line	PR1.Failed ∨ Omission-VR2_PR1	PR1.Failed ∨ Omission-VR2_PR1	PR1.Failed ∨ Omission-VR2_PR1
Omission-VF2_PF1	VF2.Failed	VF2.Failed ∨ Omission-VF1_VF2	VF2.Failed ∨ Omission-VC_VF2
Omission-VR2_PR1	VR2.Failed ∨ Omission-VR1_VR2	VR2.Failed ∨ Omission-VC_VR2	VR2.Failed ∨ Omission-VR1_VR2
Omission-VF1_VF2	VF1.Failed ∨ Omission-TF_VF1	VF1.Failed ∨ Omission-TF_VF1	No effect
Omission-VR1_VR2	VR1.Failed ∨ Omission-TR_VR1	No effect	VR1.Failed ∨ Omission-TR_VR1
Omission-TF_VF1	TF.Failed	TF.Failed	No effect
Omission-TR_VR1	TR.Failed	No effect	TR.Failed

Omission-VF1_VC	No effect	VF1.Failed ∨ Omission-TF_VF1	No effect
Omission-VC_VR2	No effect	VC.Failed ∨ Omission-VF1_VC	No effect
Omission-VR1_VC	No effect	No effect	VR1.Failed ∨ Omission-TR_VR1
Omission-VC_VF2	No effect	No effect	VC.Failed ∨ Omission-VR1_VC

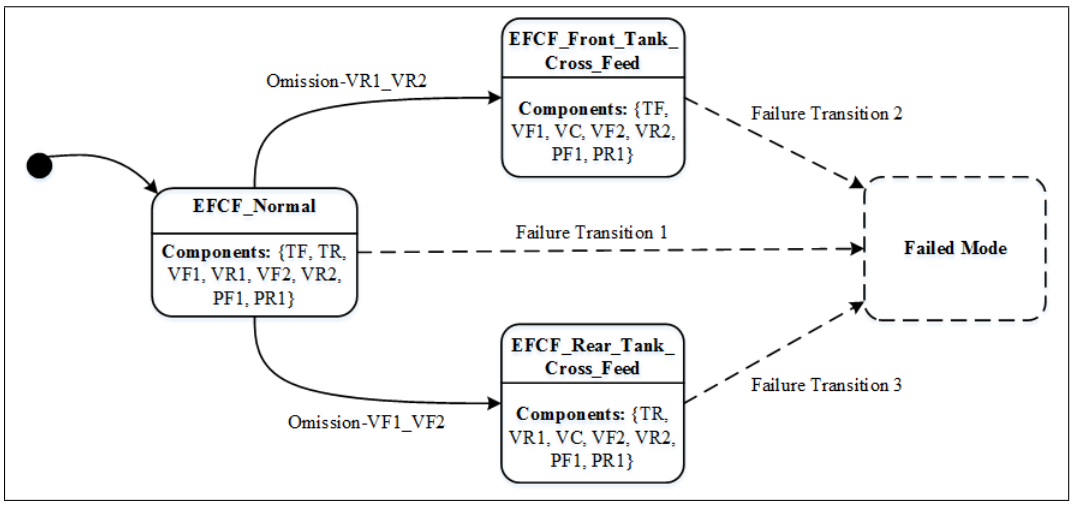


Figure 65: Functional mode chart of EFCF subsystem

Table 20: Tabular representation of mode chart

Source Mode	Transition Condition	Destination Mode
EFCF_Normal	Omission-VR1_VR2	EFCF_Front_Tank_Cross_Feed
EFCF_Normal	Omission-VF1_VF2	EFCF_Rear_Tank_Cross_Feed

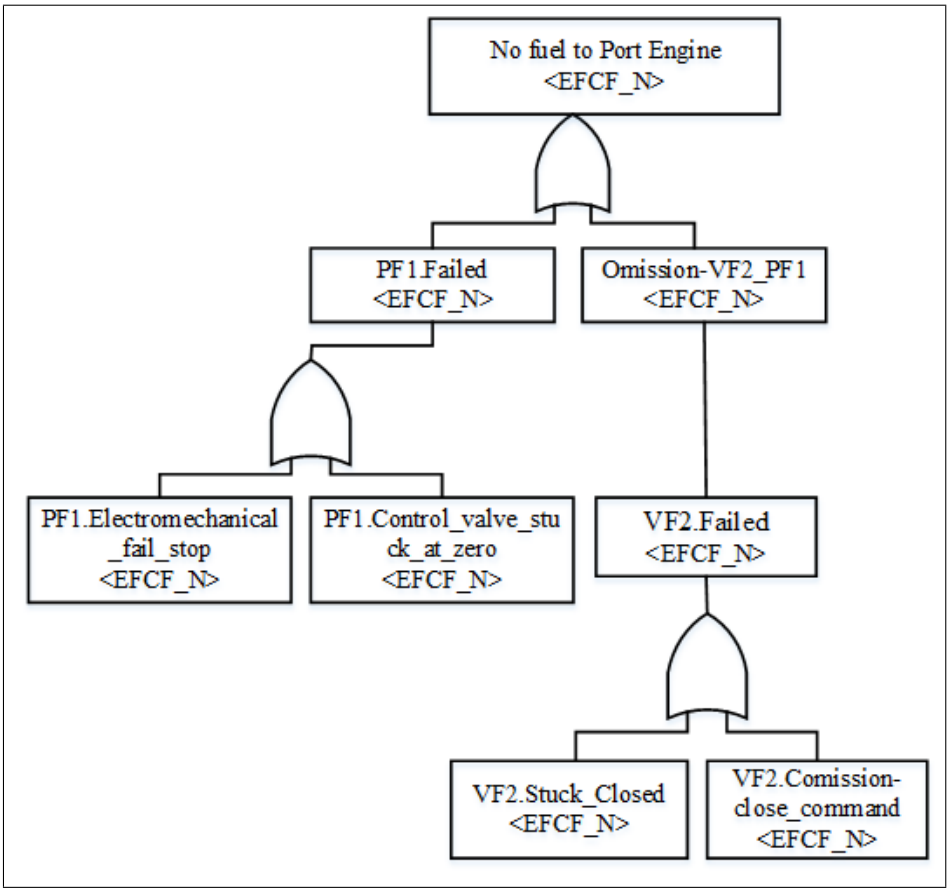


Figure 66: Fault tree for the condition “no fuel to the port engine” in EFCF_N mode

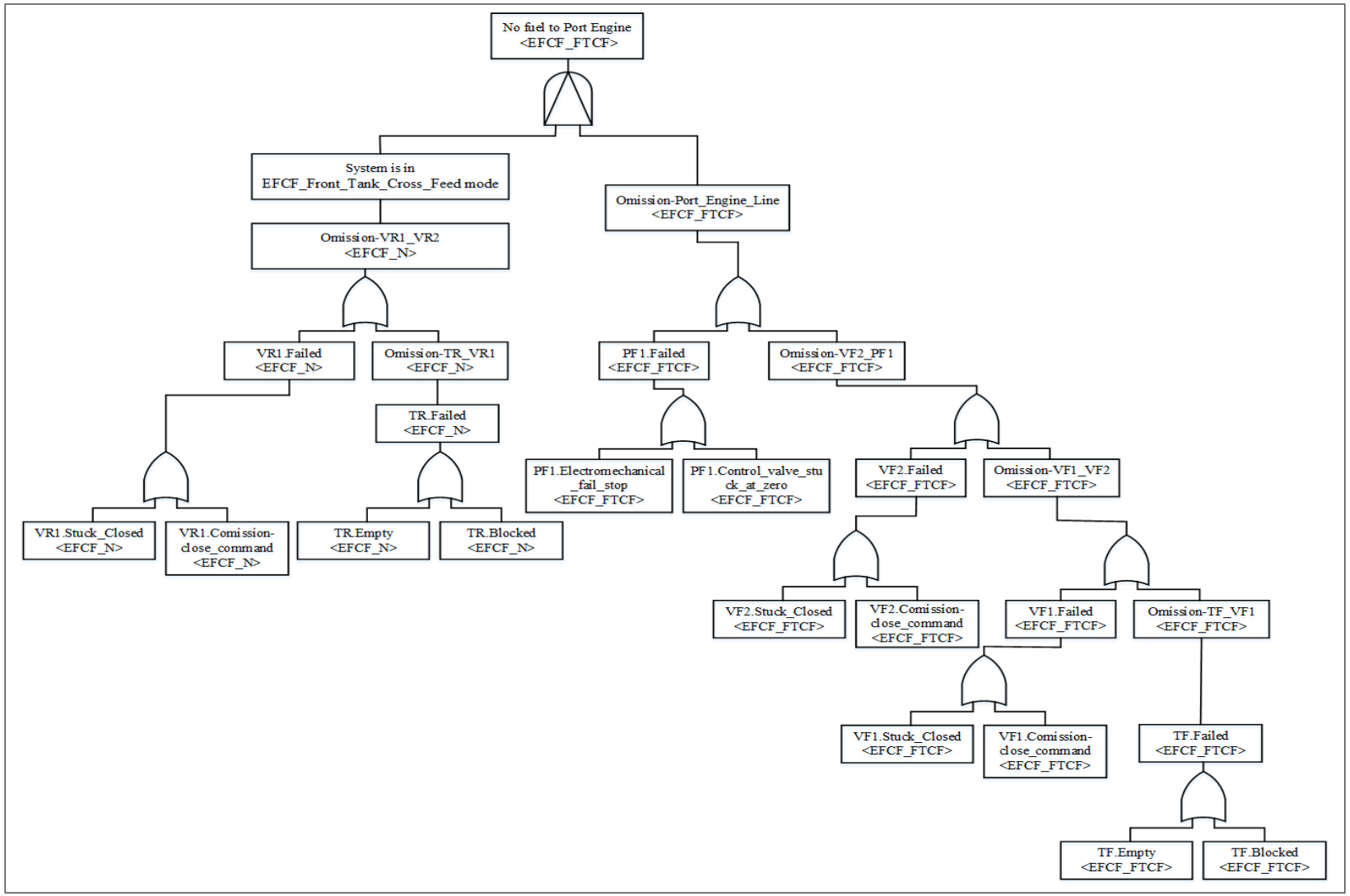


Figure 67: Temporal fault tree for the condition “no fuel to the port engine” in EFCF_FTCF mode

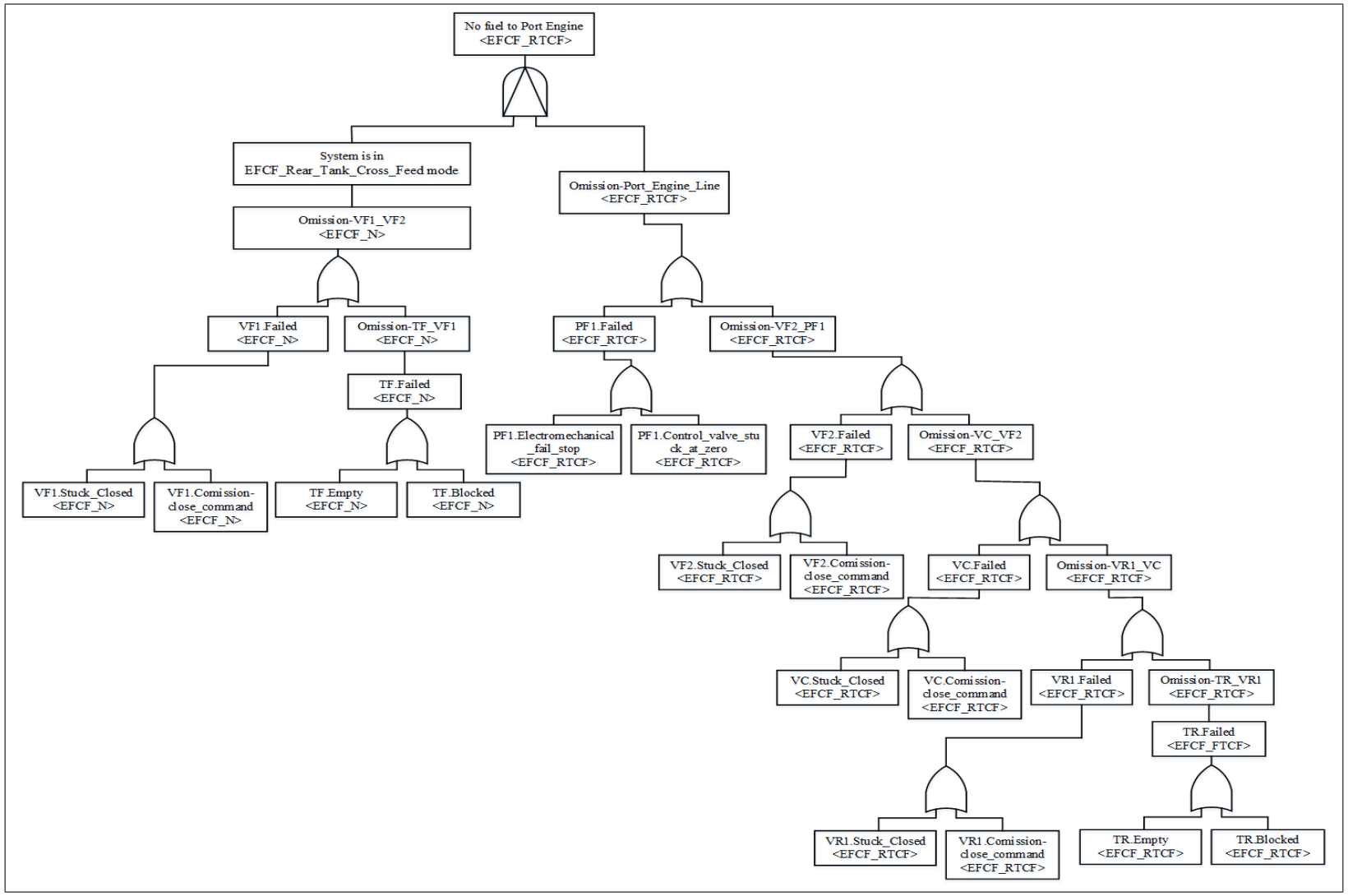


Figure 68: Temporal fault tree for the condition “no fuel to the port engine” in EFCF_RTCF mode

Table 21: Name and ID of the Basic events of the fault trees in Figures 66, 67, and 68

Event Name	Event ID	Event Name	Event ID
PF1.Electomechanical_fail_stop<EFCF_N>	X_1	TF.Blocked<EFCF_FTCF>	X_{16}
PF1.Control_valve_stuck_at_zero<EFCF_N>	X_2	VF1.Stuck_Closed<EFCF_N>	X_{17}
VF2.Stuck_Closed<EFCF_N>	X_3	VF1.Comission-close_command<EFCF_N>	X_{18}
VF2.Comission-close_command<EFCF_N>	X_4	TF.Empty<EFCF_N>	X_{19}
VR1.Stuck_Closed<EFCF_N>	X_5	TF.Blocked<EFCF_N>	X_{20}
VR1.Comission-close_command<EFCF_N>	X_6	PF1.Electomechanical_fail_stop<EFCF_RTCF>	X_{21}
TR.Empty<EFCF_N>	X_7	PF1.Control_valve_stuck_at_zero<EFCF_RTCF>	X_{22}
TR.Blocked<EFCF_N>	X_8	VF2.Stuck_Closed<EFCF_RTCF>	X_{23}
PF1.Electomechanical_fail_stop<EFCF_FTCF>	X_9	VF2.Comission-close_command<EFCF_RTCF>	X_{24}
PF1.Control_valve_stuck_at_zero<EFCF_FTCF>	X_{10}	VC.Stuck_Closed<EFCF_RTCF>	X_{25}
VF2.Stuck_Closed<EFCF_FTCF>	X_{11}	VC.Comission-close_command<EFCF_RTCF>	X_{26}
VF2.Comission-close_command<EFCF_FTCF>	X_{12}	VR1.Stuck_Closed<EFCF_RTCF>	X_{27}
VF1.Stuck_Closed<EFCF_FTCF>	X_{13}	VR1.Comission-close_command<EFCF_RTCF>	X_{28}
VF1.Comission-close_command<EFCF_FTCF>	X_{14}	TR.Empty<EFCF_RTCF>	X_{29}
TF.Empty<EFCF_FTCF>	X_{15}	TR.Blocked <EFCF_RTCF>	X_{30}

In the mode chart of Figure 65, all the functional modes are represented as a solid rounded rectangle and a failed mode is represented as a dashed rounded rectangle. Essentially the failed modes and the transitions to the failed modes from the functional modes are not a part of the functional mode chart, therefore they are shown as dashed lines in the mode chart and omitted in the tabular form. The failed mode and the transitions to the failed mode are just to show how a failed mode can be reached from different functional modes. For the subsystem EFCF, we are analysing to determine the causes of the condition "no fuel to the port engine", hence the failed mode corresponds to the output deviation `Omission-Port_Engine_Line`. Following the instructions provided in the subsection 3.4.2, the mode based fault trees for the output deviation of the subsystem EFCF are obtained and shown in Figures 66, 67, and 68 respectively. The names and the IDs of the basic events of the fault trees in Figures 66, 67, and 68 are shown in Table 21. Using the methodologies shown in section 3.4.3, the fault trees are analysed and the following 44 minimal cut sequences are obtained.

Using a prototype version of the HiP-HOPS tool the Minimal cut sequences to cause the top event ("no fuel to the port engine") are calculated to be:

1. X_1
2. X_2
3. X_3
4. X_4
5. $X_5 \lhd X_{13} \wedge X_5 \wr X_{14} \wedge X_5 \wr X_{15} \wedge X_5 \wr X_{16}$
6. $X_5 \wr X_{13} \wedge X_5 \lhd X_{14} \wedge X_5 \wr X_{15} \wedge X_5 \wr X_{16}$
7. $X_5 \wr X_{13} \wedge X_5 \wr X_{14} \wedge X_5 \lhd X_{15} \wedge X_5 \wr X_{16}$
8. $X_5 \wr X_{13} \wedge X_5 \wr X_{14} \wedge X_5 \wr X_{15} \wedge X_5 \lhd X_{16}$
9. $X_6 \lhd X_{13} \wedge X_6 \wr X_{14} \wedge X_6 \wr X_{15} \wedge X_6 \wr X_{16}$
10. $X_6 \wr X_{13} \wedge X_6 \lhd X_{14} \wedge X_6 \wr X_{15} \wedge X_6 \wr X_{16}$
11. $X_6 \wr X_{13} \wedge X_6 \wr X_{14} \wedge X_6 \lhd X_{15} \wedge X_6 \wr X_{16}$
12. $X_6 \wr X_{13} \wedge X_6 \wr X_{14} \wedge X_6 \wr X_{15} \wedge X_6 \lhd X_{16}$
13. $X_7 \lhd X_{13} \wedge X_7 \wr X_{14} \wedge X_7 \wr X_{15} \wedge X_7 \wr X_{16}$
14. $X_7 \wr X_{13} \wedge X_7 \lhd X_{14} \wedge X_7 \wr X_{15} \wedge X_7 \wr X_{16}$
15. $X_7 \wr X_{13} \wedge X_7 \wr X_{14} \wedge X_7 \lhd X_{15} \wedge X_7 \wr X_{16}$
16. $X_7 \wr X_{13} \wedge X_7 \wr X_{14} \wedge X_7 \wr X_{15} \wedge X_7 \lhd X_{16}$
17. $X_8 \lhd X_{13} \wedge X_8 \wr X_{14} \wedge X_8 \wr X_{15} \wedge X_8 \wr X_{16}$
18. $X_8 \wr X_{13} \wedge X_8 \lhd X_{14} \wedge X_8 \wr X_{15} \wedge X_8 \wr X_{16}$
19. $X_8 \wr X_{13} \wedge X_8 \wr X_{14} \wedge X_8 \lhd X_{15} \wedge X_8 \wr X_{16}$
20. $X_8 \wr X_{13} \wedge X_8 \wr X_{14} \wedge X_8 \wr X_{15} \wedge X_8 \lhd X_{16}$
21. $X_{17} \lhd X_{25} \wedge X_{17} \wr X_{26} \wedge X_{17} \wr X_{27} \wedge X_{17} \wr X_{28} \wedge X_{17} \wr X_{29} \wedge X_{17} \wr X_{30}$
22. $X_{17} \wr X_{25} \wedge X_{17} \lhd X_{26} \wedge X_{17} \wr X_{27} \wedge X_{17} \wr X_{28} \wedge X_{17} \wr X_{29} \wedge X_{17} \wr X_{30}$

23. $X_{17} \wr X_{25} \wedge X_{17} \wr X_{26} \wedge X_{17} \lhd X_{27} \wedge X_{17} \wr X_{28} \wedge X_{17} \wr X_{29} \wedge X_{17} \wr X_{30}$
24. $X_{17} \wr X_{25} \wedge X_{17} \wr X_{26} \wedge X_{17} \wr X_{27} \wedge X_{17} \lhd X_{28} \wedge X_{17} \wr X_{29} \wedge X_{17} \wr X_{30}$
25. $X_{17} \wr X_{25} \wedge X_{17} \wr X_{26} \wedge X_{17} \wr X_{27} \wedge X_{17} \wr X_{28} \wedge X_{17} \lhd X_{29} \wedge X_{17} \wr X_{30}$
26. $X_{17} \wr X_{25} \wedge X_{17} \wr X_{26} \wedge X_{17} \wr X_{27} \wedge X_{17} \wr X_{28} \wedge X_{17} \wr X_{29} \wedge X_{17} \lhd X_{30}$
27. $X_{18} \lhd X_{25} \wedge X_{18} \wr X_{26} \wedge X_{18} \wr X_{27} \wedge X_{18} \wr X_{28} \wedge X_{18} \wr X_{29} \wedge X_{18} \wr X_{30}$
28. $X_{18} \wr X_{25} \wedge X_{18} \lhd X_{26} \wedge X_{18} \wr X_{27} \wedge X_{18} \wr X_{28} \wedge X_{18} \wr X_{29} \wedge X_{18} \wr X_{30}$
29. $X_{18} \wr X_{25} \wedge X_{18} \wr X_{26} \wedge X_{18} \lhd X_{27} \wedge X_{18} \wr X_{28} \wedge X_{18} \wr X_{29} \wedge X_{18} \wr X_{30}$
30. $X_{18} \wr X_{25} \wedge X_{18} \wr X_{26} \wedge X_{18} \wr X_{27} \wedge X_{18} \lhd X_{28} \wedge X_{18} \wr X_{29} \wedge X_{18} \wr X_{30}$
31. $X_{18} \wr X_{25} \wedge X_{18} \wr X_{26} \wedge X_{18} \wr X_{27} \wedge X_{18} \wr X_{28} \wedge X_{18} \lhd X_{29} \wedge X_{18} \wr X_{30}$
32. $X_{18} \wr X_{25} \wedge X_{18} \wr X_{26} \wedge X_{18} \wr X_{27} \wedge X_{18} \wr X_{28} \wedge X_{18} \wr X_{29} \wedge X_{18} \lhd X_{30}$
33. $X_{19} \lhd X_{25} \wedge X_{19} \wr X_{26} \wedge X_{19} \wr X_{27} \wedge X_{19} \wr X_{28} \wedge X_{19} \wr X_{29} \wedge X_{19} \wr X_{30}$
34. $X_{19} \wr X_{25} \wedge X_{19} \lhd X_{26} \wedge X_{19} \wr X_{27} \wedge X_{19} \wr X_{28} \wedge X_{19} \wr X_{29} \wedge X_{19} \wr X_{30}$
35. $X_{19} \wr X_{25} \wedge X_{19} \wr X_{26} \wedge X_{19} \lhd X_{27} \wedge X_{19} \wr X_{28} \wedge X_{19} \wr X_{29} \wedge X_{19} \wr X_{30}$
36. $X_{19} \wr X_{25} \wedge X_{19} \wr X_{26} \wedge X_{19} \wr X_{27} \wedge X_{19} \lhd X_{28} \wedge X_{19} \wr X_{29} \wedge X_{19} \wr X_{30}$
37. $X_{19} \wr X_{25} \wedge X_{19} \wr X_{26} \wedge X_{19} \wr X_{27} \wedge X_{19} \wr X_{28} \wedge X_{19} \lhd X_{29} \wedge X_{19} \wr X_{30}$
38. $X_{19} \wr X_{25} \wedge X_{19} \wr X_{26} \wedge X_{19} \wr X_{27} \wedge X_{19} \wr X_{28} \wedge X_{19} \wr X_{29} \wedge X_{19} \lhd X_{30}$
39. $X_{20} \lhd X_{25} \wedge X_{20} \wr X_{26} \wedge X_{20} \wr X_{27} \wedge X_{20} \wr X_{28} \wedge X_{20} \wr X_{29} \wedge X_{20} \wr X_{30}$
40. $X_{20} \wr X_{25} \wedge X_{20} \lhd X_{26} \wedge X_{20} \wr X_{27} \wedge X_{20} \wr X_{28} \wedge X_{20} \wr X_{29} \wedge X_{20} \wr X_{30}$
41. $X_{20} \wr X_{25} \wedge X_{20} \wr X_{26} \wedge X_{20} \lhd X_{27} \wedge X_{20} \wr X_{28} \wedge X_{20} \wr X_{29} \wedge X_{20} \wr X_{30}$
42. $X_{20} \wr X_{25} \wedge X_{20} \wr X_{26} \wedge X_{20} \wr X_{27} \wedge X_{20} \lhd X_{28} \wedge X_{20} \wr X_{29} \wedge X_{20} \wr X_{30}$
43. $X_{20} \wr X_{25} \wedge X_{20} \wr X_{26} \wedge X_{20} \wr X_{27} \wedge X_{20} \wr X_{28} \wedge X_{20} \lhd X_{29} \wedge X_{20} \wr X_{30}$
44. $X_{20} \wr X_{25} \wedge X_{20} \wr X_{26} \wedge X_{20} \wr X_{27} \wedge X_{20} \wr X_{28} \wedge X_{20} \wr X_{29} \wedge X_{20} \lhd X_{30}$

First four MCSQs each represents a single point of failure. The 5^{th} MCSQ, $X_5 \lhd X_{13} \wedge X_5 \wr X_{14} \wedge X_5 \wr X_{15} \wedge X_5 \wr X_{16}$ can be interpreted as:
"The event with ID X_5 occurs before X_{13}, as long as none of X_{14}, X_{15}, and X_{16} has occurred yet."

Although the above cut sequences and the cut sets obtained from the static analysis in section 3.5.2 represent the necessary and sufficient conditions for the same top event (no fuel to the port engine), they are significantly different. The first noticeable difference is that in the static analysis the concept of mode is absent and it also does not take the sequencing among events into account while capturing the failure behaviour of the system, whereas in the dynamic analysis events are modelled as mode dependent and sequencing among the events are also considered. To understand the effect of not considering the sequencing of events in the static analysis, let us examine a minimal cut set. For instance, consider the minimal cut set 7.

MCS 7: `VF1.Stuck_Closed` $\wedge$ `VR1.Stuck_Closed`

According to the above MCS, there will be no fuel flow to the port engine if both the

valves VF1 and VR1 failed. That means it does not matter whether VF1 fails first or VR1 fails first, the only thing that matters is that failure of both them in any sequence will cause the port engine to be starved of fuel. A superficial look at the system architecture from Figure 58 would confirm that this MCS is a necessary and sufficient condition to stop fuel flow to the port engine. This is because these valves are in two possible paths through which the port engine can receive fuel, e.g., the valve VF1 is in the path from the front tank (TF) to the port engine and the valve VR1 is in the path from rear tank (TR) to the port engine. Therefore, the above MCS is initially convincing from a qualitative perspective.

Now, let us examine the quantitative aspects associated with the above MCS. In the static analysis, fixed values are used as the failure rate of the valves VF1 and VR1 (see Table 17). If we recall the system description, then we can see that under normal operating condition the port engine is fed from the front tank and the starboard engine is fed from the rear tank. In the condition of failure, the front tank can provide fuel to both the engines, similarly, the rear tank can also provide fuel to both engines. As fuel is consumed from these tanks, to maintain appropriate level of fuel in these tanks, fuel is transferred to the front tank from right wing tanks through the central tank and on the other hand the rear tank gets fuel from left wing tanks through central tank. That means the activation of the fuel flow from the left and right wing tanks as well as the amount of pressure on these tanks depend on two things.

- Which tank is supplying fuel to the engines (front tank or rear tank or both).
- How fast the fuel is consumed from the tanks.

For example, assume that the engines have a demand for m litres of fuel per minute. Therefore, in the functional condition, m litres of fuel flows through the valves VF1 and VR1. Now, if the valve VF1 failed first, event `VF1.Stuck_Closed` occurred, then the rear tank has to meet the demand of both the engines which results into double fuel flow ($2m$ litres/minute) through the VR1. On the other hand if the VR1 failed first (`VF1.Stuck_Closed` occurred) then VF1 has to allow double flow through it. That means the pressure on the valves VF1 and VR1 can be doubled depending on which of them failed first. As pressure on the valves increases, their chance of failure should also increase, i.e., failure rate should vary. However, the static analysis left out this possibility by using a fixed failure rate value and by not considering the order of event occurrence.

This is just one effect of the order of component failure on the component themselves; there may be other associated issues. For example, if VF1 fails first, then to continue the fuel flow to the port engine it is necessary to open the valve VC which may also contribute to the probability of top event (no fuel to port engine) because now fuel has to flow through three valves (first VR1, then VC, and finally VF2) whereas before the failure of VF1 fuel has to pass through two valves (first VF1, then VF2). It also increases the pressure on the rear tank which will eventually increase the pressure on the left wing

tanks, thus making the rear tank as well as the left wing tanks more susceptible to failure. On the other hand, if VR1 fails first then the path through which fuel flows to the port engine remains the same; however, now the front tank and right wing tanks become more susceptible to failure because of the increased pressure on them.

To address the above mentioned issues with the event sequencing in the static analysis, the dynamic analysis explicitly highlights the event sequencing in the minimal cut sequences and also defines mode specific failure data for the basic events. For example, MCSQ 5 and 23 as shown below (event IDs are replaced by event names) are two MCSQs where the sequencing between the events contained in MCS 7 are explicitly defined.

MCSQ 5:

VR1.Stuck_Closed<EFCF_N> $\lhd$ **VF1.Stuck_Closed<EFCF_FTCF>**
$\wedge$ VR1.Stuck_Closed<EFCF_N> $\wr$ VF1.Comission_close_command <EFCF_FTCF>
$\wedge$ VR1.Stuck_Closed<EFCF_N> $\wr$ TF.Empty<EFCF_FTCF>
$\wedge$ VR1.Stuck_Closed<EFCF_N> $\wr$ TF.Blocked<EFCF_FTCF>

MCSQ 23:

VF1.Stuck_Closed<EFCF_N> $\wr$ VC.Stuck_Closed<EFCF_RTCF>
$\wedge$ VF1.Stuck_Closed<EFCF_N> $\wr$ VC.Comission_close_command <EFCF_RTCF>
$\wedge$ **VF1.Stuck_Closed<EFCF_N>** $\lhd$ **VR1.Stuck_Closed<EFCF_RTCF>**
$\wedge$ VF1.Stuck_Closed<EFCF_N> $\wr$ VR1.Comission_close_command <EFCF_RTCF>
$\wedge$ VF1.Stuck_Closed<EFCF_N> $\wr$ TR.Empty <EFCF_RTCF>
$\wedge$ VF1.Stuck_Closed<EFCF_N> $\wr$ TR.Blocked <EFCF_RTCF>

To explicitly highlight the change in failure rate of same component, mode specific failure rates are defined. For example, in MCSQ 23, VF1.Stuck_Closed<EFCF_N> and in MCSQ 5, VF1.Stuck_Closed<EFCF_FTCF> represent the failure of VF1. Although they represent the failure of the same component, due to their mode tag (i.e., the terms inside the angle brackets) they are two different events with different failure data. More specifically, VF1.Stuck_Closed<EFCF_N> represents the failure of VF1 when the system is in normal condition, i.e., front tank is providing fuel to the port engine only. On the other hand, VF1.Stuck_Closed<EFCF_FTCF> represent the failure of VF1 when front tank is proving fuel to both the engines, i.e., VF1 has double flow through it. Similarly, VR1.Stuck_Closed<EFCF_N> is failure of VR1 when it has normal flow (*m* litres/minute) through it and VR1.Stuck_Closed<EFCF_RTCF> is the failure of VR1 when double fuel flows through it. Consequently, each also has a different failure rate defined to reflect this different workload.

Chapter 4

Quantitative Dependability Analysis under Uncertainty

4.1 Introduction

The previous chapter describes how state-based annotations can be added to the architectural model of dynamic systems. This allows state machines (SMs) representing the dynamic failure behaviour of the systems to be obtained by synthesising the system model using approaches like HiP-HOPS, and subsequently the SMs can be translated into Pandora temporal fault trees using the algorithm proposed by Mahmud *et al.* (2012). Pandora temporal fault trees will produce qualitative information about the failure behaviour of the system as a set of minimal cut sequences. However, to get numerical information regarding the dependability of the systems, e.g., probability of the system being in a non-operational state after a certain amount of time, or by identifying the most critical components, a quantitative analysis is required.

Section 2.2.2.5 shows that HiP-HOPS can quantify Pandora temporal fault trees using an analytical approach based on the failure rate of components. However, this assumes that the exact failure rates of components are always available. Therefore, HiP-HOPS is unable to perform quantitative analysis under uncertainty, i.e., if the failure rates of components are uncertain then it cannot provide any quantitative information about the dependability of the systems. If this limitation in performing quantitative analysis under uncertainty can be overcome, by redefining the analytical operators of the gates used in Pandora TFT to incorporate uncertainty in failure rates of components or by translating Pandora TFTs into other models capable of making decision under uncertainty, then HiP-HOPS can be successfully used for uncertainty analysis in a dynamic dependability analysis.

This chapter presents the basic concepts of quantitative analysis in HiP-HOPS and proposes three methodologies for improved analysis of Pandora TFTs, thus improving the efficacy of MBSA approaches to perform quantitative analysis under uncertainty.

4.2 Quantitative Dependability Analysis of Dynamic Systems

The quantitative analysis of Pandora TFT is performed based on the result of the qualitative analysis to help estimate the probability of the top event occurring from the given failure rates of basic events. As the top event of a TFT is represented as a disjunction of a set of minimal cut sequences (MCSQs), first the probabilities of all the MCSQs are calculated using the formulas defined for Boolean and temporal gates in Section 2.2.2.2 and 2.2.2.5 respectively based on the presence of the gates in the expressions of the MCSQs. All the dynamic analyses performed in HiP-HOPS are based on some assumptions about the behaviour of the components failure events. Firstly, events are considered as non-repairable, i.e., the event is false as long as the fault has not occurred and when the fault occurs, the event become true and will remain true afterward. In other words, events can only go from false (non-failed) to true (failed) but not vice versa. It is also assumed that an event goes instantly from false to true.

Secondly, basic events are considered to be statistically independent. For example, assume X and Y are two events with failure probability $Pr\{X\}$ and $Pr\{Y\}$. If they are considered as dependent then probability of both events occurring is:

$$Pr\{X \wedge Y\} = Pr\{X\} \cdot Pr\{Y|X\} = Pr\{Y\} \cdot Pr\{X|Y\} \tag{18}$$

Where $Pr\{Y|X\}$ is the conditional probability of Y given X meaning that the occurrence of Y may be affected by the occurrence of X and vice versa.
However, similar to classical FTA, HiP-HOPS considers that all basic events are independent; therefore, the probability of both events occurring is:

$$Pr\{X \wedge Y\} = Pr\{X\} \cdot Pr\{Y\} \tag{19}$$

The analytical solution uses failure rates or occurrence probability in a continuous time domain and exponential distribution of the failure rates are used. In the analytical solution, as the time is considered as continuous and events as statistically independent, the simultaneous occurrence of two events is considered to have null probability, therefore the SAND gate is usually excluded (as the probability of a MCSQ containing a SAND gate equates to zero) from the quantitative analysis.

Apart from the above assumptions, compositional MBSA approaches like HiP-HOPS do not consider any uncertainty aspect in the failure rates or failure probabilities of the basic components. As mentioned earlier, the integrity of the results obtained from quantitative analysis completely depends on the data used in the analysis. Sometimes it is difficult to find precise failure rates of components due to the scarcity of statistical data, or complex architecture of systems, or change in operational environment, or using completely new components. Moreover, in the early design phase if no decision has been

made about the components that are going to be used in the system then failure rates of components cannot be decided. Therefore, if uncertain failure rate data are used as the basis of the quantitative analysis in approaches like HiP-HOPS without addressing the uncertainty, then the uncertainty at the basic event level will propagate towards the top event, and thus make the outcomes uncertain as well. Although, an analytical technique is available to probabilistically quantify Pandora temporal fault trees, it is only applicable to systems which have components with exponentially distributed failure data. For this reason, it is worth exploring the alternative options to provide a state space solution to Pandora TFTs.

In this thesis, Petri Nets, Bayesian Networks and fuzzy set theory have been identified as three potential approaches to extend the capabilities of compositional MBSA approaches by addressing uncertainty in the quantitative analysis of dynamic systems. The next three sections will describe the methodologies to quantify temporal fault trees based on Petri Nets (PNs), Bayesian Networks (BNs) and fuzzy set theory. The Petri Net based approach is for a continuous time domain and is based on the exponential distribution of failure rates of components. The fuzzy set theory based methodology is based on the analytical solution of the Pandora TFTs; therefore all assumptions of the analytical solution are left unchanged. The BNs based approach is for an interval-based discrete time domain and this approach can work with any types of component lifetime distributions.

4.3 A Petri Net based Methodology

The primary goal of the Petri Net based methodology is to translate the Pandora temporal fault trees into generalised stochastic Petri Nets (GSPN), thus providing a state-space solution to TFTs. This section defines a method for translating Pandora temporal fault trees into GSPNs. During the transformation process, all the assumptions of the Pandora temporal fault trees are taken into account. In the proposed mapping strategy, each temporal fault tree node (basic, intermediate, and top events) is mapped to a sub-net where there is a place indicating the status of the node. We use places to represent the state of the system, timed transitions to represent random faults and immediate transitions to represent failure propagation. A place is graphically represented as a circle, a timed transition is represented as a white bar and an immediate transition is represented as a black bar. Timed transitions are characterised by the failure rate of basic events and in this thesis, for the PN based method, we use exponential distribution of failure rates.

The mapping of each TFT gate to GSPN should be correct in the sense that in the sub-net there is a place representing the outcome of the gate and if all the conditions are fulfilled for the gate outcome to be true then the place gets a token. As we use the exponential firing rates (failure rates) for the timed transitions, two transitions firing at the same time is zero. For this reason, if we transform the SAND gate containing two or more basic events into a Petri Net then during the whole mission time we will not get any

token in the place representing the SAND output, i.e., probability of the SAND outcome is always 0. For this reason, during quantitative analysis using exponential failure rates we can ignore any MCSQ that contains a SAND gate and therefore there is no need to transform a SAND gate to PN. The process of transforming all the TFT gates except the SAND gate is described in the following subsections. The logical correctness of all the mappings of TFT gates to Petri Nets is verified by constructing and testing them in the CPN tool (Jensen *et al.*, 2007).

4.3.1 Mapping of Basic Events to GSPN

This section describes the process of transforming a Pandora basic event to a Petri Net model. During the transformation process we have to make sure that all the assumptions about the basic events in the Pandora TFTs remain unchanged. The mapping of a basic event to a Petri Net is shown in Figure 69.

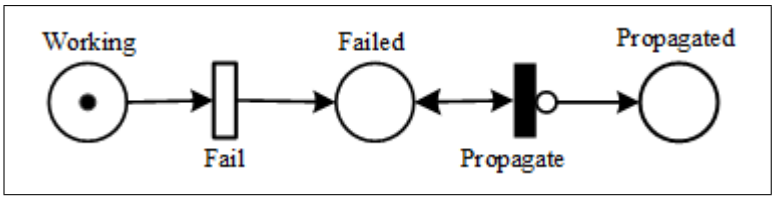


Figure 69: Mapping of a basic event to a Petri Net

In the PN model of a basic event, the condition (occurred or non-occurred) of the event is determined by the presence or absence of a token in some particular place. In the general case of PN model, a place can have multiple tokens; however, in this case, a place can have at most one token. This is due to the fact that the places are now representing the status of components which could either be true (occurred) or false (non-occurred) and can essentially be represented by the presence and absence of a single token. As all components are assumed to be fully functional at time 0, the place (*Working*) representing the fully functional state has a token. Every component is considered to have an exponentially distributed failure rate, and the occurrence of an event (component failure) is represented by the timed transition named *Fail*. Once this transition fires, the event occurs, i.e., the component corresponding to the event has failed.

Usually, when a transition fires in a PN model a pre-specified number of tokens are removed from the input place(s) of the transition and a pre-specified number of tokens are deposited to the output place(s) of the transition. However, in the case of a basic event, the amount (percentage) of tokens that are removed from the place *Working* are deposited to the place *Failed* and this amount is determined by the transition *Fail* which is characterised by the exponential distribution of the failure rate of the component. For example, if the firing rate of transition *Fail* is set as unity (1) then on firing of this transition a token would be consumed from the place *Working* and a token would be deposited in the place *Failed*. However, failure rates of components are usually far less than 1, so the

firing rate would not be unity. Depending on the failure rate, a fraction of the token would be consumed from the place *Working* and accumulated in the place *Failed*. For example, if the failure rate of a component is $0.0001/hour$ then the probability that the component will fail after 1000 hours would be 0.095. Now, for this component in the PN model the transition *Fail* would be set to fire at the rate of $0.0001/hour$. Table 22 shows the process of token removal and accumulation process for this component for a duration of 1000 hours by setting the discretisation step as 100 hours.

Table 22: Token removal and accumulation process in the PN model of a basic event

Time (hours)	**Token in the places**	
	Working	**Failed**
0	1.00	0
100	0.990	0.010
200	0.980	0.020
300	0.970	0.030
400	0.961	0.039
500	0.951	0.049
600	0.942	0.058
700	0.932	0.068
800	0.923	0.077
900	0.914	0.086
1000	0.905	0.095

So the place *Failed* represents the failure state of a component. Now, we have to model the failure propagation for the event. In Pandora, a failure is propagated instantly, i.e., there is no delay between the occurrence and the propagation of the failure. For this reason, an immediate transition named *Propagate* is used to propagate the failure (see Figure 69). The outgoing arrow from transition *Propagate* back to place *Failed* serves to maintain persistency of events, i.e., maintain the permanent failure state of the event irrespective of any further propagation. To ensure that the error is propagated exactly once in a single path, the inhibitor arc from place *Propagated* to transition *Propagate* is used.

4.3.2 Mapping of the Boolean Gates to GSPN

Mapping of the Boolean AND and OR gates to a Petri Net is based on the work of Bobbio *et al.* (1999). An AND gate can have any number of input events connected to it and the outcome of the AND gate becomes true iff all the input events to it become true. The mapping of an n input AND gate to a PN model is shown in Figure 70.

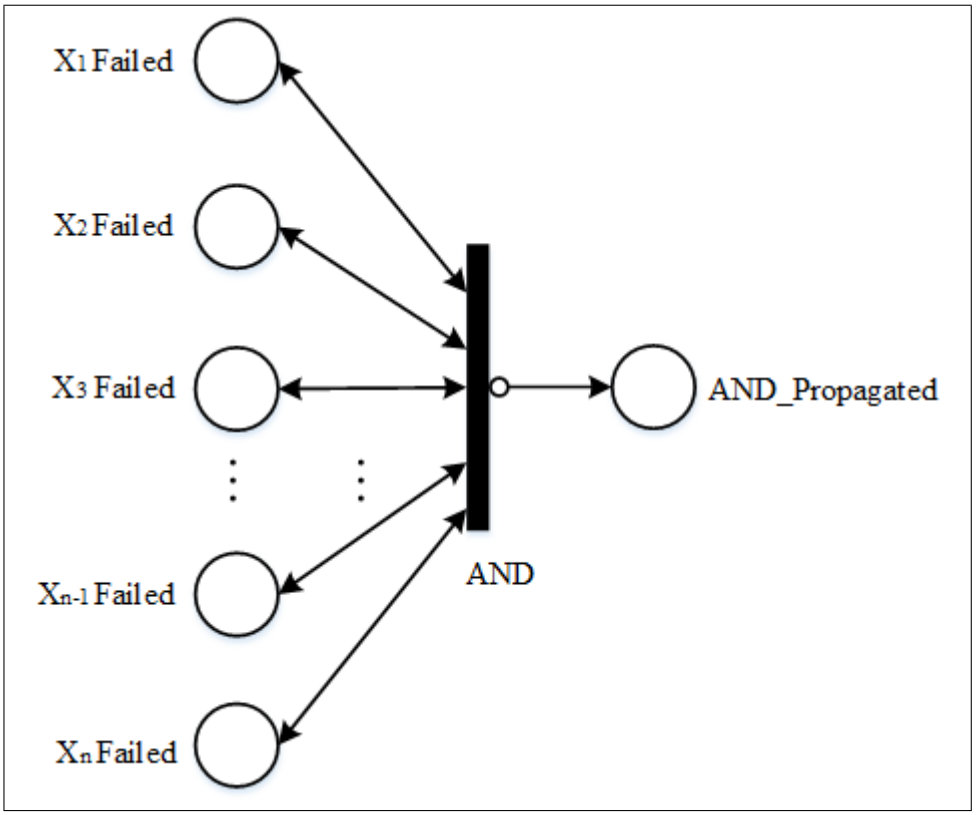


Figure 70: Mapping of an *n* input AND gate to Petri Net

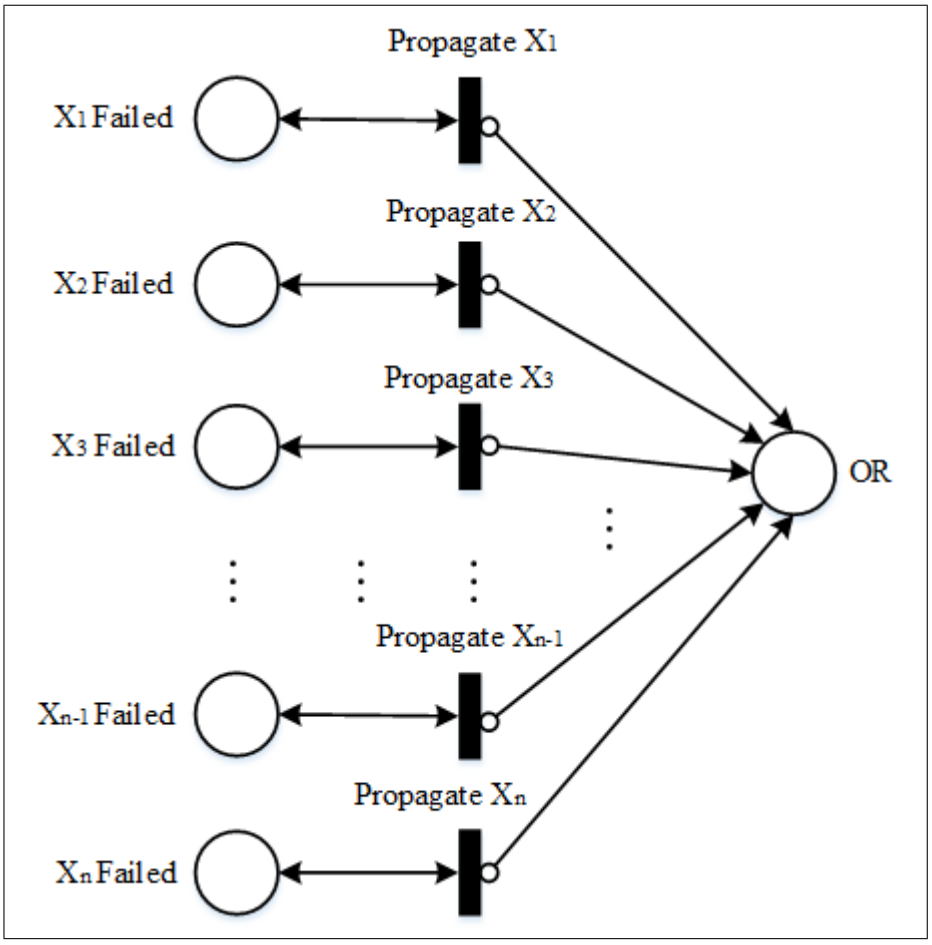


Figure 71: Mapping of an *n* input OR gate to Petri Net

The places X_1 *Failed*, X_2 *Failed*, X_3 *Failed*, $\cdots$, X_{n-1} *Failed*, and X_n *Failed* represent the input events to the AND gate. All the places are connected to a single immediate transition (*AND*) with bidirectional arrows, therefore the transition will fire when all the input places have a token (i.e., all input events occur) and on firing it deposits a token to the place representing the outcome of the AND gate (*AND_Propagated*).

Unlike the AND gate, the OR gate represents a logical disjunction of the input events, and means the outcome of the OR gate becomes true if at least one of the input events become true. The OR gate is translated to a PN model by creating a transition for each place corresponding to input events of the OR gate (see Figure 71). Hence, as soon as one of the input places gets a token, one of the transitions will fire and deposit a token to the place corresponding to the outcome of the OR gate (*OR*). In both the cases (AND and OR) no token is consumed from the input places because an event may be the input of several gates. For this reason, the bi-directional arrows are used to connect the places to the transition(s).

4.3.3 Mapping of the Temporal Gates to GSPN

The mapping of the temporal gates to PN models is not as straightforward as that of the Boolean gates. This is due to the fact that the Boolean gates are stateless in the sense that they do not need to remember the order of occurrence of the events. However, along with the occurrence of the input events, the temporal gates must also remember the order of occurrence of input events. Now, let us recall the behaviour of Pandora's temporal gates from subsection 2.2.2.5. In a PAND gate, if there are *n* input events (X_1 *Failed*, X_2 *Failed*, X_3 *Failed*, $\cdots$, X_{n-1} *Failed*, and X_n *Failed*) then to make the PAND outcome true all the events must occur and they must occur in order from left to right, i.e., X_1 *Failed* occurs first, then X_2 *Failed*, then X_3 *Failed*, $\cdots$, then X_{n-1} *Failed*, and finally X_n *Failed*. The transformation of an *n* input PAND gate to a PN model is shown in Figure 72.

The transformation is done in such a way that ensures the exact sequencing of events to make the PAND outcome true. The place X_1 *Failed* is connected to the transition T_1 with a bi-directional arrow and all other places corresponding to other input events are connected to this transition using inhibitor arcs. As a result, the transition T_1 will fire when there exists a token in place X_1 *Failed* and no token in any other connected places (X_2 *Failed* to X_n *Failed*) . Once this transition fires, the place P_1 would get a token and essentially will represent the scenario when only the left-most input of the PAND gate occurs but not the others. Similarly, the transition T_2 will fire to deposit a token to place P_2 when the second input becomes true (i.e., place X_2 *Failed* gets a token) and all the subsequent inputs remain false. In a similar fashion, the firing of transition T_{n-1} deposits a token to place P_{n-1}, thus representing that the $(n-1)^{th}$ event to the PAND gate has occurred and the n^{th} event is still to occur. So, the presence of a token in each of the places from P_1 to P_{n-1} confirms that events X_1 *Failed* to X_{n-1} *Failed* have occurred in a sequence from left to right and the event X_n *Failed* is yet to occur.

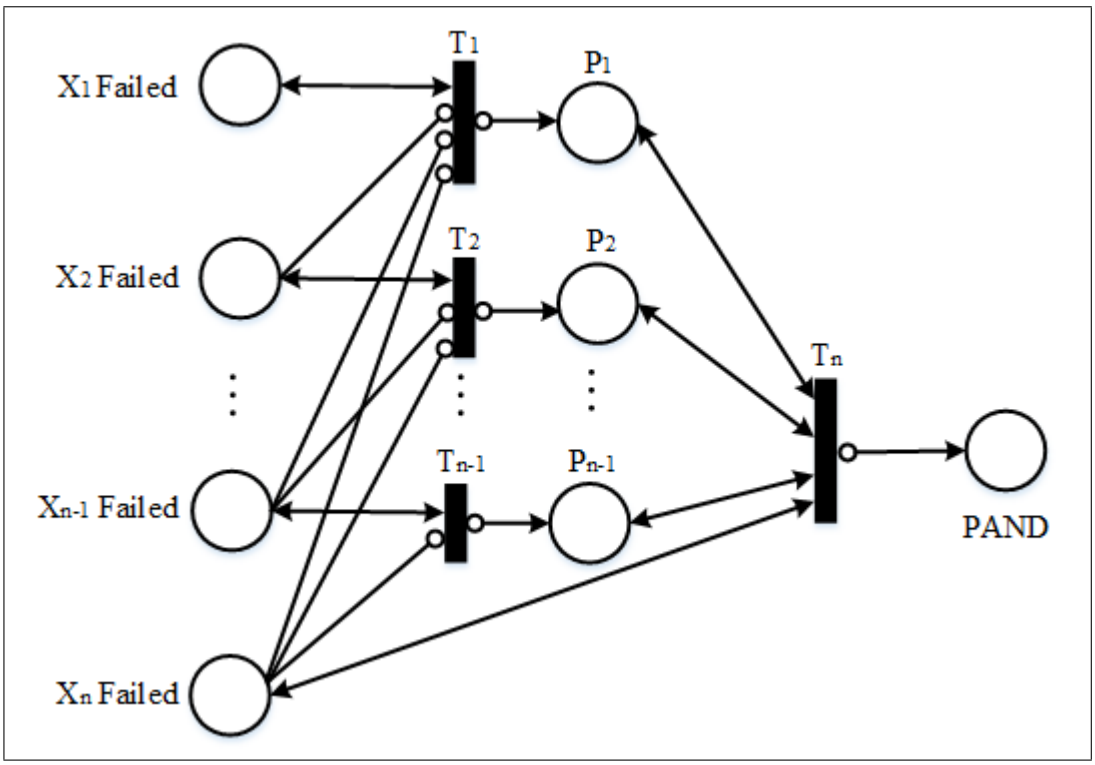


Figure 72: Mapping of an n input PAND gate to Petri Net

Now, the places P_1 to P_{n-1} and the place X_n *Failed* are connected to an immediate transition T_n and the firing of this transition would deposit a token to the place *PAND* which essentially represents the scenario when the PAND outcome becomes true, i.e., all input events to the PAND gate occurred and they occurred obeying the sequence. If a PAND gate has only two inputs then we just need to ensure sequencing between two events. The PN model of a two input PAND gate is shown in Figure 73.

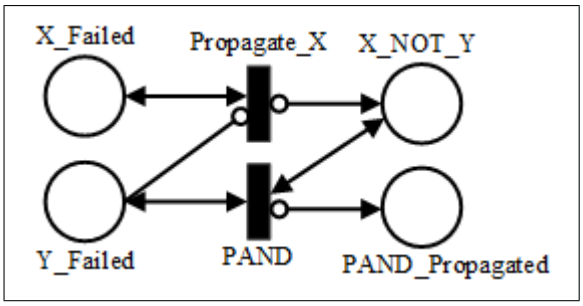


Figure 73: Mapping of a two input PAND gate to Petri Net

Similar to the PAND gate, the POR gate also defines a sequence. It is used to specify that one input event has priority and must occur first for the POR to be true, but unlike PAND does not require all other input events to occur as well. That means if the disjunction of the non-priority events occurs at all then it should occur after the priority event. For example, let us consider that X_1, X_2, X_3, $\cdots$, X_{n-1}, and X_n are n different input events to a POR gate where the event X_1 is the priority event. The outcome of the POR gate can become true if at least one of the following two conditions becomes true.

1. **Condition 1:** $X_1 \wedge \overline{(X_2 \vee X_3 \vee \cdots \vee X_{n-1} \vee X_n)}$

2. **Condition 2:** $X_1 \lhd (X_2 \vee X_3 \vee \cdots \vee X_{n-1} \vee X_n)$

The first condition refers to a scenario where only the priority event occurs. The second condition represents a scenario where one or more non-priority events may have occurred but they occurred after the priority event. The exact logical mapping of an n input POR gate to a PN is shown in Figure 74. In the figure, the place P_1 represents the logical OR of the non-priority events. The places P_2 and P_3 represent the conditions 1 and 2 respectively. The place named *POR* represents the logical OR of the two conditions, i.e., the outcome of the POR gate.

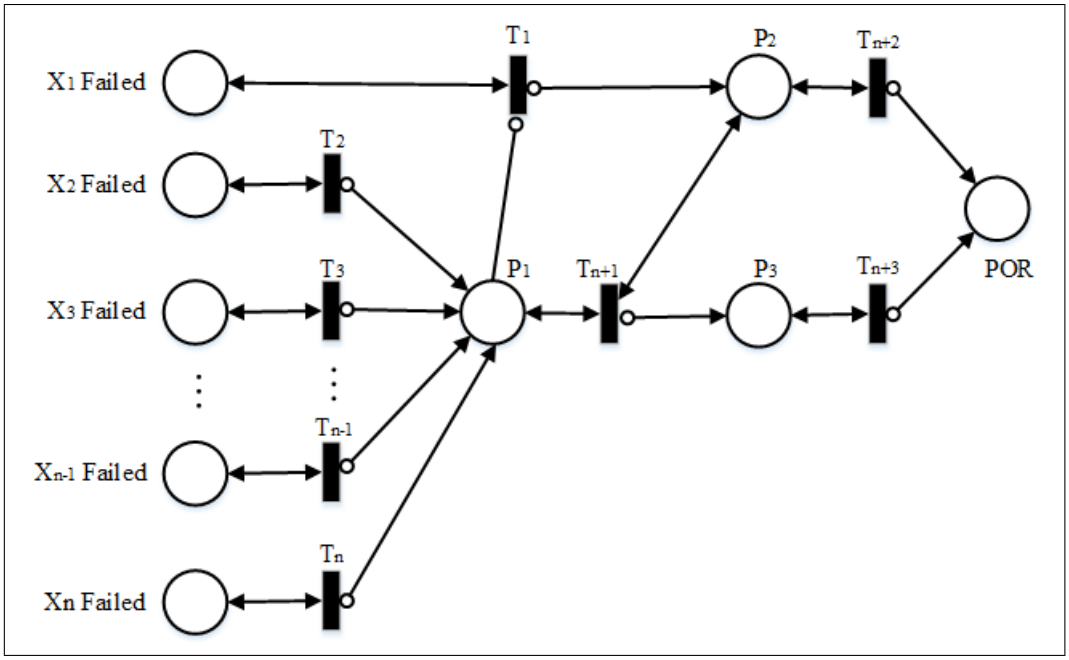


Figure 74: Mapping of an n input POR gate to Petri Net

Condition 2 actually represents a two input PAND gate where the left input is the priority events of the POR gate and the right input is the logical OR of the all non-priority events of the POR gate. From the PN model of the two input PAND gate (see Figure 73) we can see that the transition *PAND* would be enabled only if both the places X_NOT_Y and Y_Failed have a token each. For both the places to have a token each, the place X_NOT_Y must have the token first otherwise it will never have a token. That means, in terms of timing, $X \wedge \overline{Y}$ followed by Y will make the PAND outcome true. For this reason, in Figure 74, the place P_2 will always get a token first and subsequently enables the transition T_{n+2} to deposit a token to the place *POR*. So, the POR outcome will never become true because of a token in the place P_3. As a result, the place P_3 becomes redundant in the PN model, which in effect makes the transitions T_{n+1} and T_{n+3} extraneous as well. The refined PN model for the n input POR gate is shown in Figure 75. If a POR gate has two input events then Figure 75 would be reduced to Figure 76.

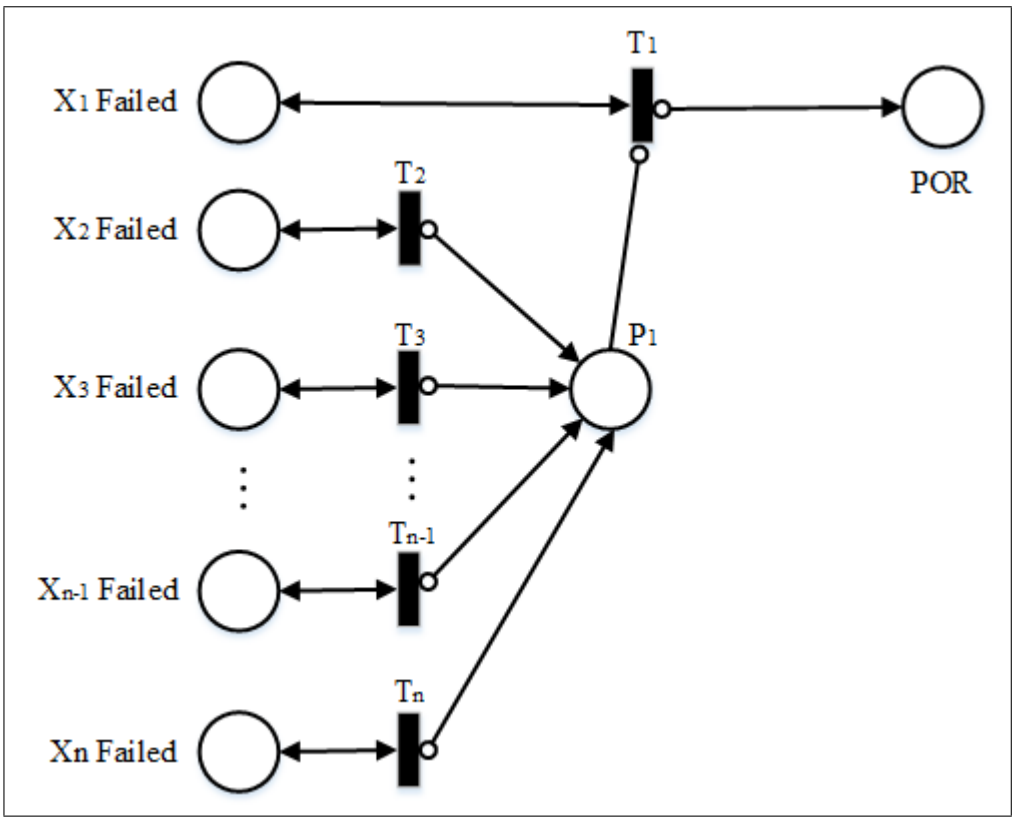


Figure 75: Refined PN model of *n* input POR gate

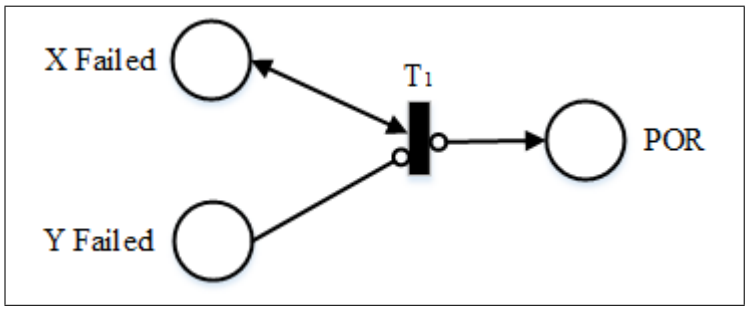


Figure 76: Mapping of a two input POR gate to PN model

4.3.4 Evaluating System Unreliability using PN based method

In Pandora TFTs, the top event (system failure) of the temporal fault tree is represented as the logical OR of the minimal cut sequences (MCSQs). A MCSQ represents the logical relation among the basic events using Boolean AND gate and the temporal gates, i.e., a MCSQ can contain any number of temporal and Boolean AND gates to combine any number of basic events. In order to probabilistically evaluate Pandora TFTs via PN models to obtain information about system unreliability, at first we need to transform the MCSQs to the PN model. After that, to represent the top event, a PN model of an OR gate must be formed by considering the places representing the outcome of the MCSQs as input. The basic events are transformed to a PN model according to the process shown in section 4.3.1 and TFT gates are transformed to PN model following the process shown in sections 4.3.2 and 4.3.3 respectively.

Once the PN model of the TFT is created, the model can be simulated using any PN simulator to obtain information about the system unreliability after a pre-specified mission time. In this thesis, an open source PN analyser named ORIS (Horváth *et al.*, 2012) is used to simulate the PN models. During simulation we need to define the mission time to decide how long the simulation will run and also have to set the step size. Step size represents the fraction of the mission time that would be covered in a single iteration. So the step size is obtained by dividing the mission time by the number of steps in which we want to finish the simulation. That means the biggest step size could be equal to the mission time and there is no restriction on the minimum step size. However, the smaller the step size the longer the simulation will take to finish.

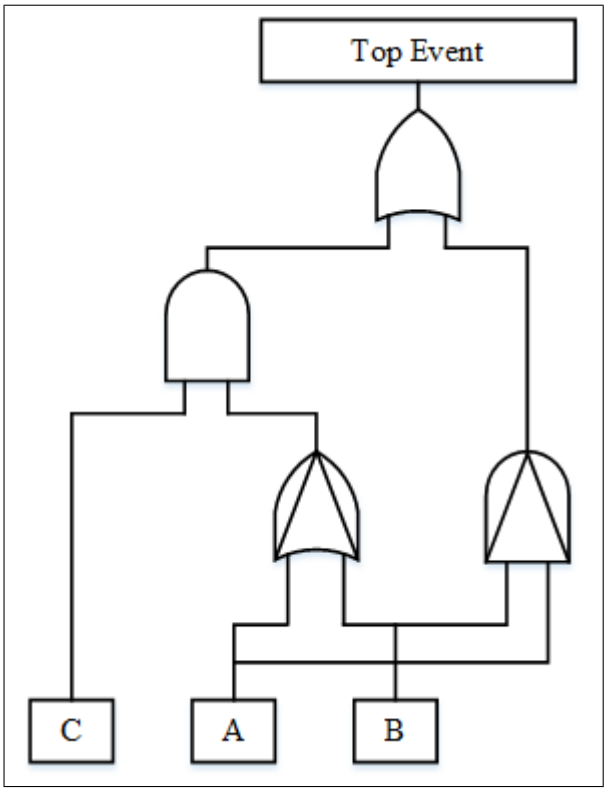


Figure 77: An example TFT

For illustration, let us consider the TFT of Figure 77. The TFT consists of three basic events (A, B, and C), one POR, one PAND, one AND, and one OR gate. The expression of the top event can be written as:

$$Top\ Event = (A \wr B) \wedge C \vee B \lhd A$$

Let us assume that the failure rate per hour of the components A, B, C are $0.0007, 0.0005$, and 0.0009 respectively. The TFT of Figure 77 is mapped into the PN model, and shown in Figure 78. All the places corresponding to the basic events have a token each at the beginning of the simulation and the places are connected to the timed transitions characterised by failure rate of the respective component. Now, if we want to know the probability of the system failure after 1000 hours, then we have to run the simulation by setting the mission time as 1000 hours. We have run the simulation for 1000 hours with 100

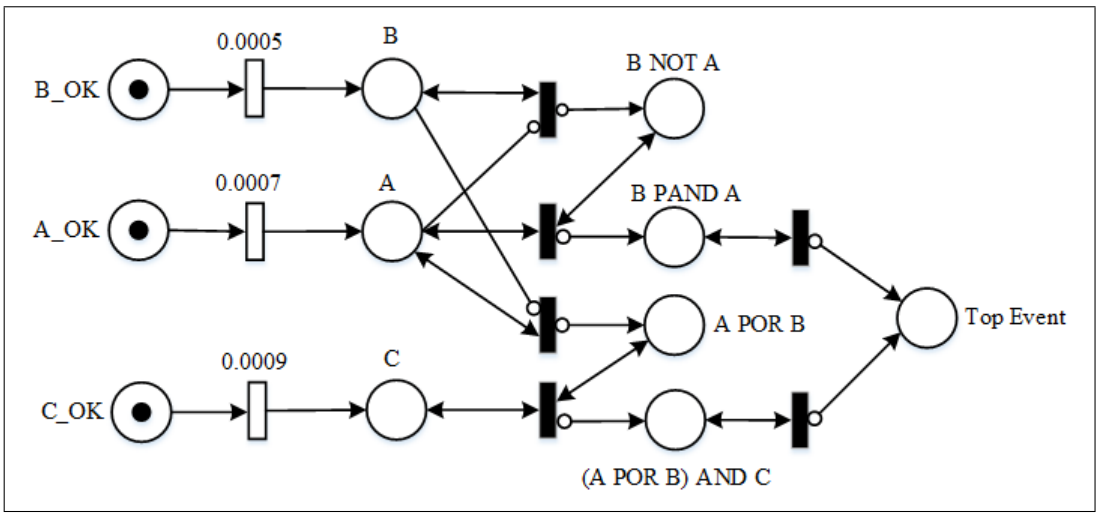


Figure 78: PN model of the TFT of Figure 77

hours step size, and the system unreliability value obtained was 0.318. From this value, a decision could be made that with the given failure rates of the components there is 31.8% chance of system failure after 1000 hours.

4.3.5 Importance Measures using the PN based Method

Importance measures determine the various contributions of basic or intermediate events to the occurrence of the top event or how a change in any of these events can affect the occurrence of the top event. This information can be served as a useful source of data for resource allocation (upgrade, maintenance etc.) and helps stakeholders in improving system dependability (safety, reliability, availability etc.). At present, HiP-HOPS can perform Fussell-Vesely and Birnbaum importance measures (Vesely *et al.*, 2002) as a part of quantitative analysis based on the fixed failure rates of the components.

Using the PN based method, we can calculate the importance of a component (or basic event) by taking the difference between the top event probability and the top event probability with the component fully available. The idea is to see the changes in the value of top event probability in the condition that the component has not failed during the mission time, i.e., when the component does not contribute at all to the top event. The probability of the top event is obtained during the system unreliability evaluation process as shown in the previous section. In a PN model, to make a component fully available, we simply need to remove the token from the place representing the component. Once we remove the token from a place, then even though the place is connected to a timed transition, during the simulation the transition will never trigger, thus the component corresponding to the place will not contribute to the occurrence of the top event. Let $Pr\{TE\}$ be the probability of the top event and $Pr\{TE|BE_{i=0}\}$ be the probability of the top event given that the basic event BE_i is fully available. So the importance measure (IM) for a basic event BE_i is defined as:

$$IM(BE_i) = Pr\{TE\} - Pr\{TE|BE_{i=0}\} \quad (20)$$

The bigger the value of the $IM(BE_i)$ the higher the importance of the component is, and all the components are ranked according to their importance. For example, the results of the importance measure calculation from the PN model of Figure 78 for the basic events of the TFT of Figure 77 are shown in table 23.

Table 23: Importance ranking for the basic events of PN in Figure 78

Basic Events (BE_i)	**Pr{TE}**	**Pr{TE|$BE_{i=0}$}**	**IM (BE_i)**	**Rank**
A	0.318	0	0.318	1
B		0.299	0.019	3
C		0.048	0.270	2

4.4 A Bayesian Networks based Methodology

The primary goal of the Bayesian network based methodology is to translate the Pandora temporal fault trees into Bayesian Networks, and thus increase the capability of MBSA techniques to perform quantitative analysis of dynamic systems under uncertainty. More importantly, it will allow the MBSA approaches to use any distributions of failure data of components in quantitative analysis, not just exponential distribution.

4.4.1 Representation of time in Pandora TFTs

In order to convert TFTs into BNs, it useful to first understand how Pandora represents time in its sequencing among events. Pandora makes few assumptions about the model of time in any system; time can be discrete or continuous, point or interval-based, or some hybrid. However, whichever way the time is represented, the model of time must be linear, i.e., branching of time is not permitted in Pandora. As the events in Pandora are persistent and occur instantly, there are only three possible temporal relation between two events: before, simultaneous, and after. For this reason, in Pandora "the exact time at which an event occurs is not important — the only thing that matters is when it occurs relative to the other events, i.e., which comes first, which comes second, which comes last etc." (Walker, 2009, p.105).

Therefore, sequence values — an abstraction of the relative time at which an event occurs — are used in Pandora instead of a quantitative, absolute metric of time. If an event has not occurred during the mission time then that event is given a sequence value 0. If an event has occurred at any time during the system operation within the mission time then it is given a sequence value greater than 0 to indicate when it occurred relative to other events, i.e., all sequence value greater than 0 represents logical true, but the

higher the value the later the events occurred. For example, sequence value 1 means the event occurred first, 2 means it occurred second and so on. If two or more events occur simultaneously, then they will have the same sequence value. As these sequence values combine the truth state (true/false) with the relative time of occurrence for each event, these values make up the basis of temporal truth tables used in Pandora. A temporal truth table showing the sequence values for each gate in Pandora given the sequence values of each input event was shown in Table 3 in section 2.2.2.5.

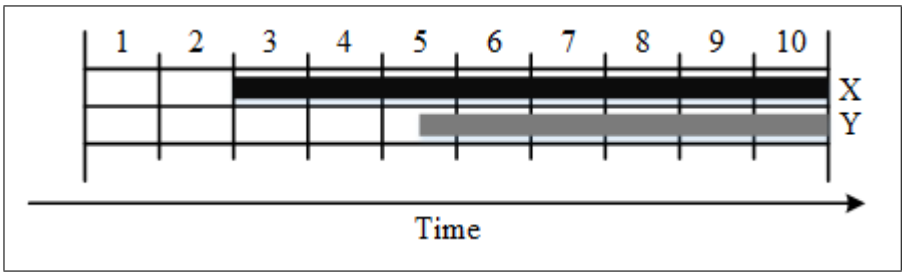


Figure 79: An interval-based Discrete time model in Pandora

An example of an interval-based discrete model of time in Pandora is shown in Figure 79. In this model, the mission time is divided into 10 equal intervals and are used to determine relative position of event occurrence, i.e., to decide sequence values for events. It is important to note that the interval numbers themselves do not directly represent the sequence values. Sequence values are assigned to events in an increasing order based on their relative occurrence time. For example, if an event *X* occurs in interval *i* and gets a sequence value *k* then if the next event *Y* occurs in any interval greater than *i* then it would be given the sequence value *k+1* because of its relative position to event *X*. As seen in the figure, once an event occurs (e.g. a component fails) it continues to hold its failed status to be persistent. In terms of deciding relative positions (before, after, or simultaneous) between events, what we actually need to know are the intervals when the events originally occurred. It does not really matter at what exact time within an interval the events occur. As seen in the figure, the event *X* occurs at the very beginning of interval 3 and event *Y* occurs in the middle of interval 5, however they will be given the sequence value 1 and 2 respectively, and this is sufficient to decide that the event *X* occurs before event *Y*. In the interval-based model of time if two events occur in the same interval but in two different times, they still would get the same sequence value, and are thus consequently considered as occurring simultaneously.

4.4.2 Conversion of Temporal Fault Trees into Bayesian Networks

In the PN based methodology proposed in section 4.3, a continuous model of time was used. For the BN based methodology, an interval-based discrete model of time is used instead. So the first step in the BN based reliability evaluation method is to translate the Pandora TFT into a discrete-time Bayesian Network. In this BN, each root node represents a basic event of the TFT and each intermediate node represents a gate (including

both Boolean and temporal gates) of the TFT. The prior probability tables (PPTs) for each of the root nodes are then defined, while the intermediate nodes must each have a conditional probability table (CPT) defined. The conversion of the TFT to BN is a simple one-to one mapping from basic events to root nodes and from logic gates to intermediate nodes in which the original fault tree connections are preserved in the form of parent/child relationships. The PPTs for the root nodes are populated based on the failure probability distributions of the basic events and the CPTs for the intermediate nodes are defined based on the behaviour of the gates used in the temporal fault trees.

As the outcome of each temporal gate is dependent on the relative sequence values of the events involved in those gates, mission time T is divided into n equal intervals from $t = 0$ to $t = T$ (see Figure 80), where n must be at least equal to the maximum order of the minimal cut sequences. The order of a minimal cut sequence (MCSQ) defines the number of basic events contribute to that MCSQ. Each time interval represents a possible non-zero sequence value, during which an event may go from its not occurred (false) state to its occurred (true) state. At the beginning of the system operation, a component is considered as fully available therefore given a sequence value 0 (i.e., the component is in a state named *State 0* (False)) and this value remains the same until the component fails, i.e., if the component can survive till the end of the mission time then it will continue carrying the sequence value 0. If a component fails in interval 1, then it will have the sequence value 1, but if the component fails in any other interval i where $1 < i \leq n$ then it can have any sequence value in between 1 to i based on its relative position to its immediate predecessor.

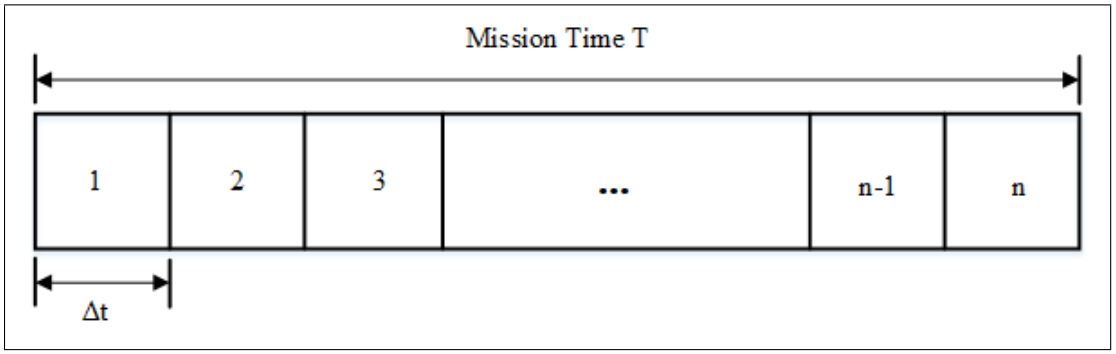


Figure 80: Division of mission time into n intervals

An event having a sequence value *'i'* is considered to be in *'State i'* and it has an associated probability value representing the probability of the event being in *'State i'*. As the root nodes in the BN represent different basic events, we need to define prior probability tables for the root nodes, where each entry in a prior probability table of a node represents the probability of the respective event being in a particular state. The values of the entries of the PPT are calculated from the failure rates of the components and the calculation process depends on the distributions of the failure rates.

If exponential distribution of failure rate is used, then we can obtain the probability of

a component being failed in the interval $[t_1, t_2]$ by integrating the probability density function (PDF) of exponential distribution, $\lambda e^{-\lambda t}$, in the following way:

$$Pr\{BE_i\}_{[t_1,t_2]} = \int_{t_1}^{t_2} \lambda e^{-\lambda t} dt \tag{21}$$

The Weibull distribution is one of the most popular component life distributions used in reliability engineering (O'Connor, 2011). The Weibull PDF is given by:

$$f(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} e^{-\left(\frac{t}{\alpha}\right)^{\beta}} \tag{22}$$

where α is scale parameter and β is shape parameter. For Weibull distribution, we can obtain the component failure probability in the interval $[t_1, t_2]$ as:

$$Pr\{BE_i\}_{[t_1,t_2]} = \int_{t_1}^{t_2} f(t) dt \tag{23}$$

Once prior probability tables for the nodes representing the basic events are created, we need to create the conditional probability tables for the nodes representing the TFT gates. Note that in Pandora the output of any gate can be either true or false but every true value is linked with a sequence value, i.e., the relative time at which the gate outcome becomes true. Therefore, the probability of an intermediate node representing a TFT gate being in a certain state given the states of its parent nodes can either be 0 or 1. For instance, let us consider a 2 input POR gate. For simplicity, we divide the mission time T into 2 intervals, i.e., n=2. This yields three possible states for each event: State 0, in which an event does not occur at all during the mission time T; State 1, in which the event occurs in the interval $[0, T/2]$, and State 2, in which the event occurs in the interval $[T/2, T]$. Each state is then associated with a probability, namely the probability of the event being in that state.

Figure 81 illustrates this example and shows the 2 input POR gate and its BN model, where the events are independent. In the BN, nodes X and Y represent the events X and Y and prior probability tables for these nodes are generated based on the failure rate of the component failure modes X and Y respectively. The arcs from nodes X and Y to node $(X\ POR\ Y)$ show the dependency of the output of the POR gate on its input events X and Y. The conditional probability table shows the logical specification of the POR gate, i.e., its output is true if its priority (left-most) input event occurs and no other events occur before or at the same time as the priority event. A 1 in the State 0 column of the CPT means the output of the POR gate is false — i.e., it is 100% probable that the output of the POR gate will not become true during the mission time T given that the input events are on those particular states. Similarly, a 1 in the State 1 column means that it is 100% probable that the POR output will become true in interval 1 given those particular inputs, and the same for State 2 and interval 2.

For instance, we can say that the probability that the POR gate will be in State 0 given

that event X is in State 1 and event Y is State 0 is 0, i.e., $Pr\{(X\ POR\ Y) = State\ 0 \mid X = State\ 1,\ Y = State\ 0\} = 0$. This is as we would expect, since the POR gate will have the same sequence value as its priority event if the left-most (priority) event X occurs and the other event Y does not occur. Instead, the POR outcome should be in State 1, i.e., $Pr\{(X\ POR\ Y) = State\ 1 \mid X = State\ 1, Y = State\ 0\} = 1$. This can also be seen in the temporal truth table in Table 3: if X is 1 and Y is 0, then $(X\ POR\ Y)$ should also be 1.

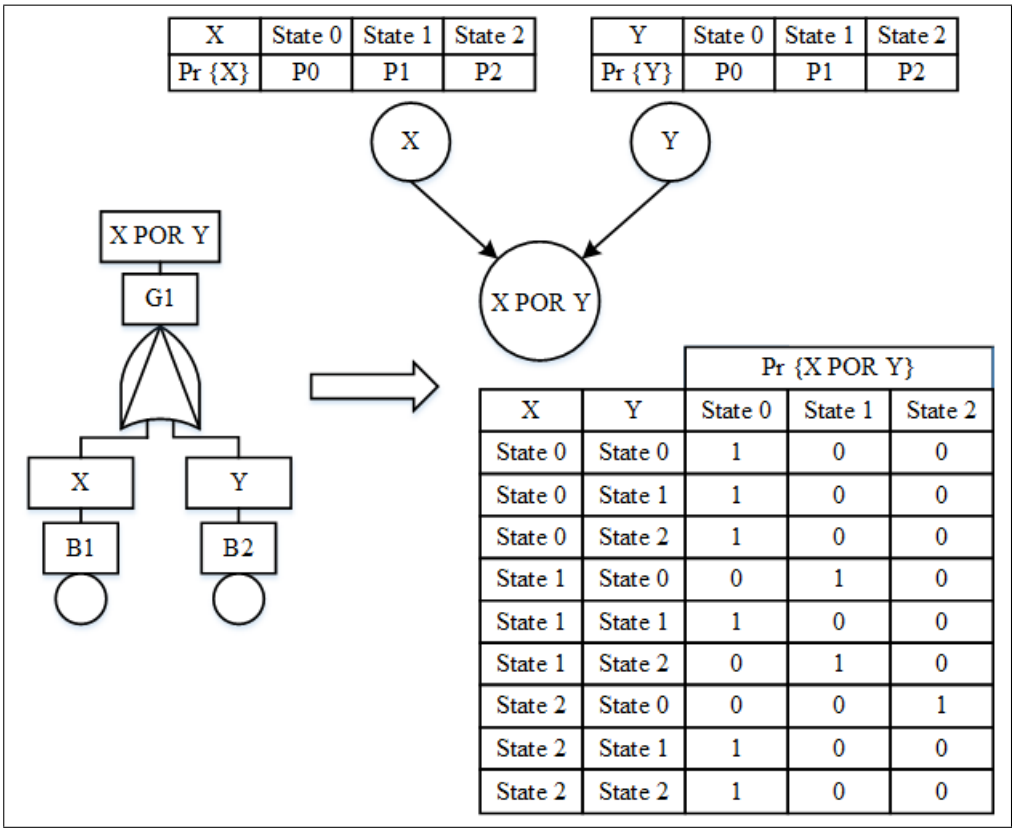


Figure 81: Two input POR gate and its BN model

The mapping of a two input PAND gate with *n=2* is shown in Figure 82. Similar to the POR gate the CPT of the PAND gate resembles its temporal truth table. As seen in the CPT of the PAND gate, the PAND outcome becomes true only in one scenario when the first input (X) is in State 1 and the second input (Y) is in State 2, i.e., they occur in a sequence from left to right. In this case, the state associated with the PAND outcome is State 2 because this is the state when the last input of the PAND becomes true.

As mentioned earlier a MCSQ can contain any number of gates from PAND, POR, and AND gate, and the top event is represented as the logical OR of the MCSQs. So a MCSQ combines the basic events using both temporal and Boolean gates. Due to the behavioural specifications, the temporal gates have to consider the state (sequence number) of the events and the Boolean gates do not need to consider the state information. However, in a MCSQ, the outcome of an AND gate may be connected to the input of a temporal gate, and vice versa. So, to maintain the compatibility between the outputs and the inputs we need to define temporal behaviour for the AND gate. The mapping of a two input AND gate to a BN model by dividing the mission time into two equal intervals is shown in

Figure 83. The behaviour of the AND gate is as such that its outcome becomes true only when all of its inputs become true, and it does not impose any condition on the sequencing of events. From the CPT of the AND gate in Figure 83, we can see that the AND outcome becomes true (1's in column State 1 or State 2) when the input events are either in State 1 or State 2. If any of the input event is in State 0 (logical false) then the outcome of the gate is false (1's in the column State 0). So the CPT of the PAND, POR, and the AND are of the same pattern and entries in the table are either 0 or 1, but the positions of the 0's and 1's change according to the logic of the gates.

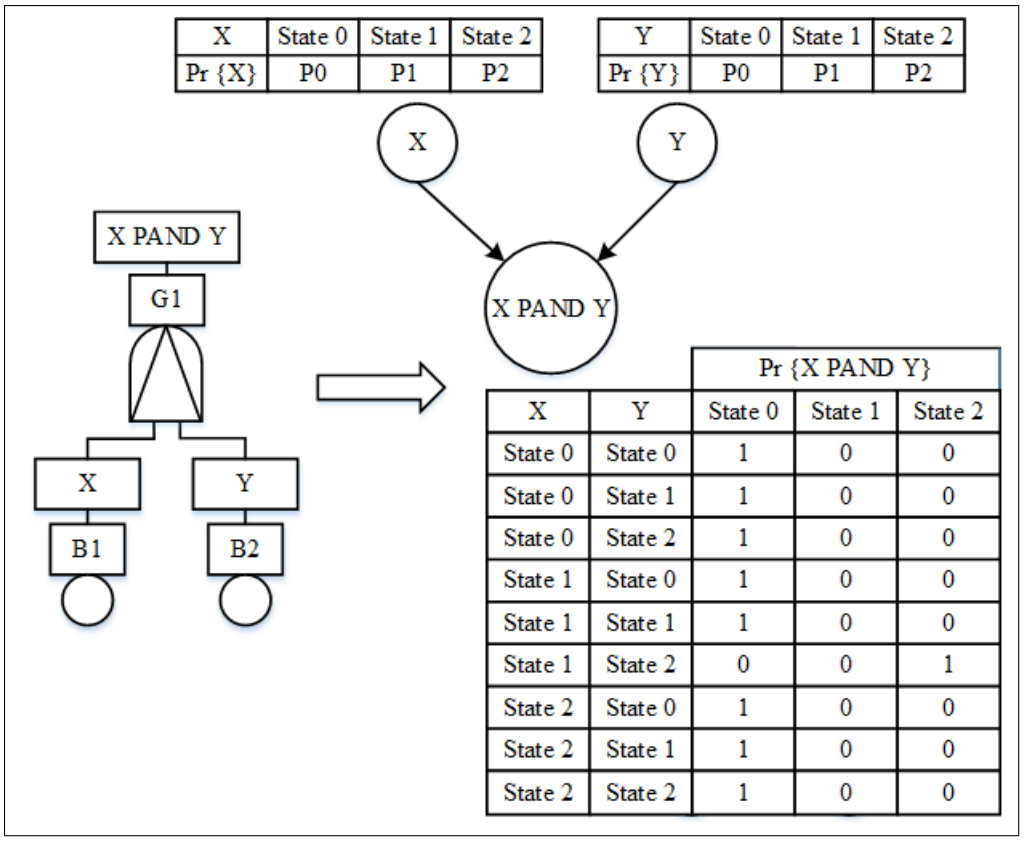
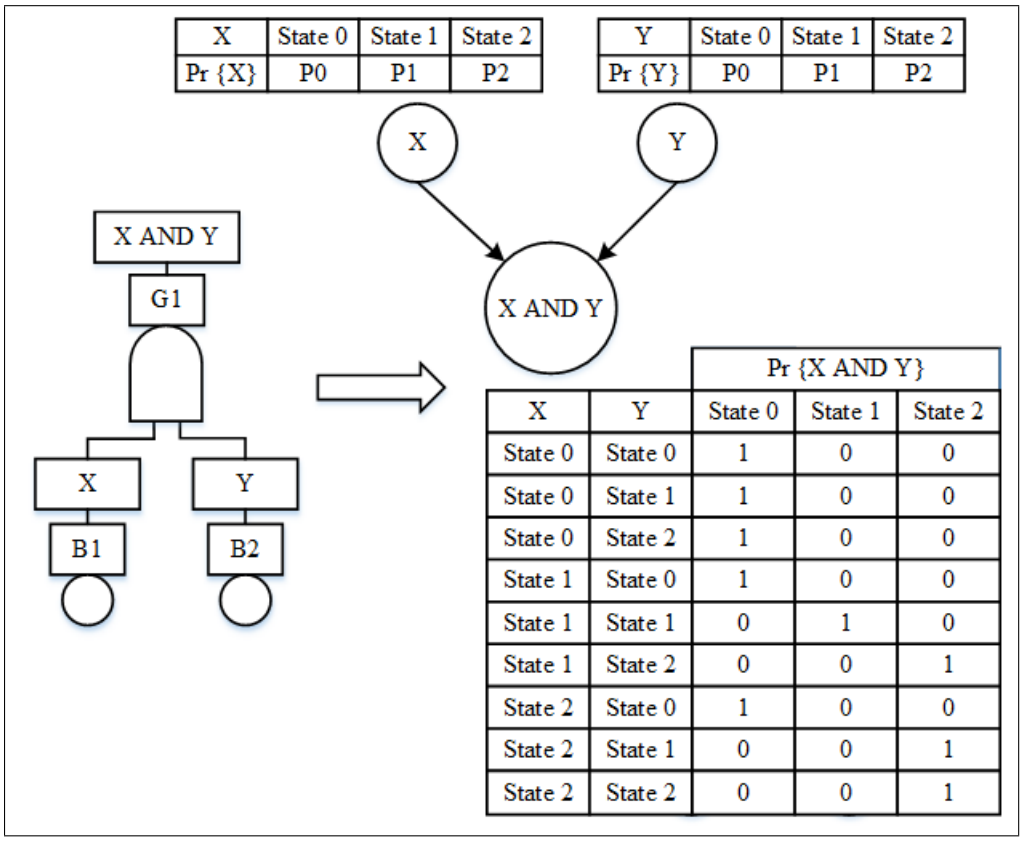


Figure 82: Two input PAND gate and its BN model

Now comes the mapping of the Boolean OR gate to the BN. In Pandora, the OR gate is used to combine the MCSQs to represent the top event of the TFTs. So in a minimised Pandora TFT, there is only one OR gate. As the primary aim of the quantitative evaluation of Pandora TFT is to estimate the system unreliability, i.e., probability of the top event being true (system failure) after a specified time, unlike the AND gate the Boolean representation of the output of the OR gate would suffice. One thing to note that the inputs to the OR gate are the MCSQs with temporal information (with states) and the output of the OR gate is either true or false, so a mapping from temporal input to Boolean output is needed. Mapping of a two input OR gate to a BN model is shown in Figure 84, where each of the input events has three possible states. It is seen from the CPT of the OR gate that the probability of the OR outcome being true is 100% if at least one of its input is true (either in State 1 or State 2).

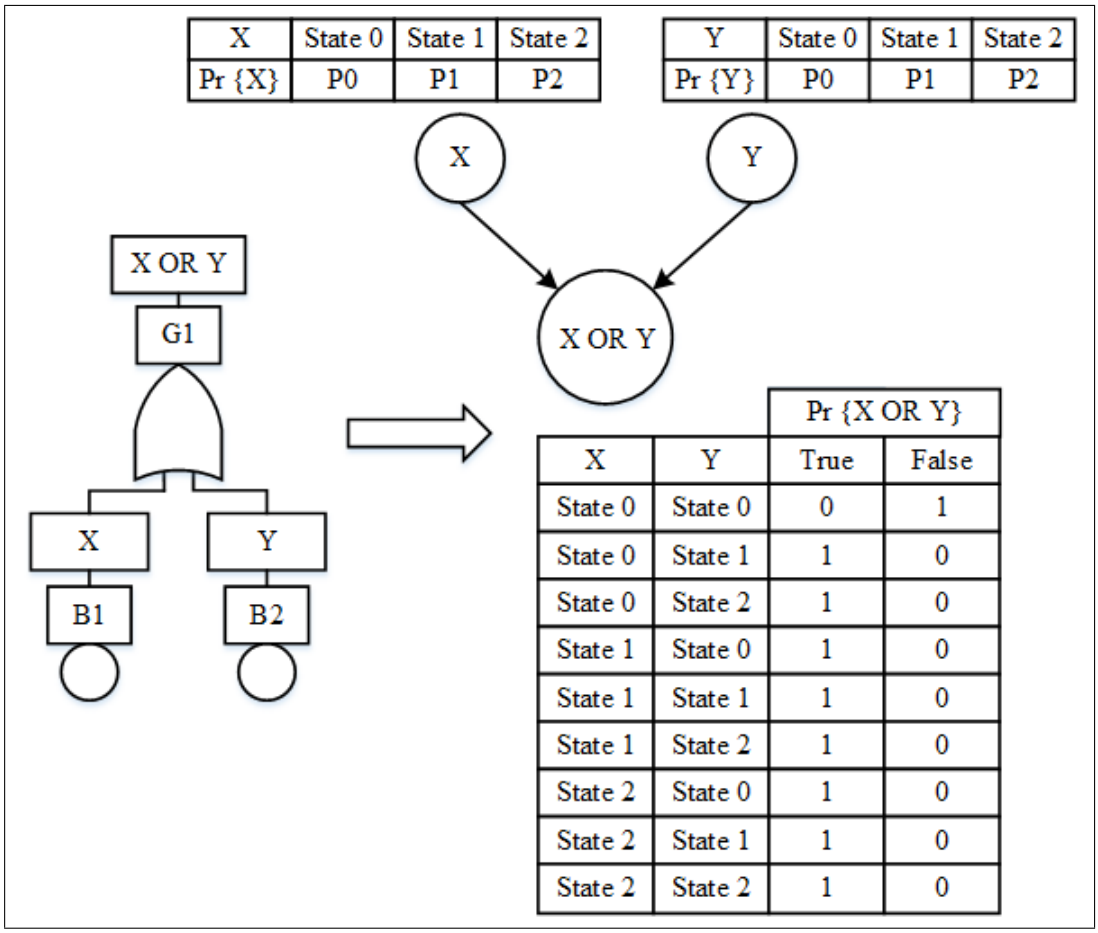


X	State 0	State 1	State 2
Pr {X}	P0	P1	P2

Y	State 0	State 1	State 2
Pr {Y}	P0	P1	P2

		Pr {X AND Y}		
X	Y	State 0	State 1	State 2
State 0	State 0	1	0	0
State 0	State 1	1	0	0
State 0	State 2	1	0	0
State 1	State 0	1	0	0
State 1	State 1	0	1	0
State 1	State 2	0	0	1
State 2	State 0	1	0	0
State 2	State 1	0	0	1
State 2	State 2	0	0	1

Figure 83: Two input AND gate and its BN model



X	State 0	State 1	State 2
Pr {X}	P0	P1	P2

Y	State 0	State 1	State 2
Pr {Y}	P0	P1	P2

		Pr {X OR Y}	
X	Y	True	False
State 0	State 0	0	1
State 0	State 1	1	0
State 0	State 2	1	0
State 1	State 0	1	0
State 1	State 1	1	0
State 1	State 2	1	0
State 2	State 0	1	0
State 2	State 1	1	0
State 2	State 2	1	0

Figure 84: Two input OR gate and its BN model

4.4.3 Evaluating System Unreliability using the BN based method

Once we have the Pandora TFT of the failure behaviour of a system, we can translate the TFT to BN model and subsequently perform a query on the BN model to obtain the system unreliability. We have modified an open source tool named JavaBayes (Cozman, 2001) to create the BN model and to perform the query on it. For illustration, the example TFT of Figure 77 is translated into BN model, shown in Figure 85. In the BN, the root nodes are named after their corresponding basic events in the TFT. Prior probability tables for the root nodes are populated based on the failure rate of the events ($0.0007, 0.0005$, and 0.0009 for events A, B, and C respectively) and for 1000 hours of mission time. Table 24 shows the prior probability values for the basic events for $n=5$ (number of intervals the mission time is divided into). The CPT for each of the inner nodes of the BN is generated automatically based on the logical specification of the gate the node is representing. After that, to obtain the probability of the top event of the TFT, a query is performed on the node named *Top Event*, and the value obtained is 0.305. This value represents the probability of the system failure after 1000 hours.

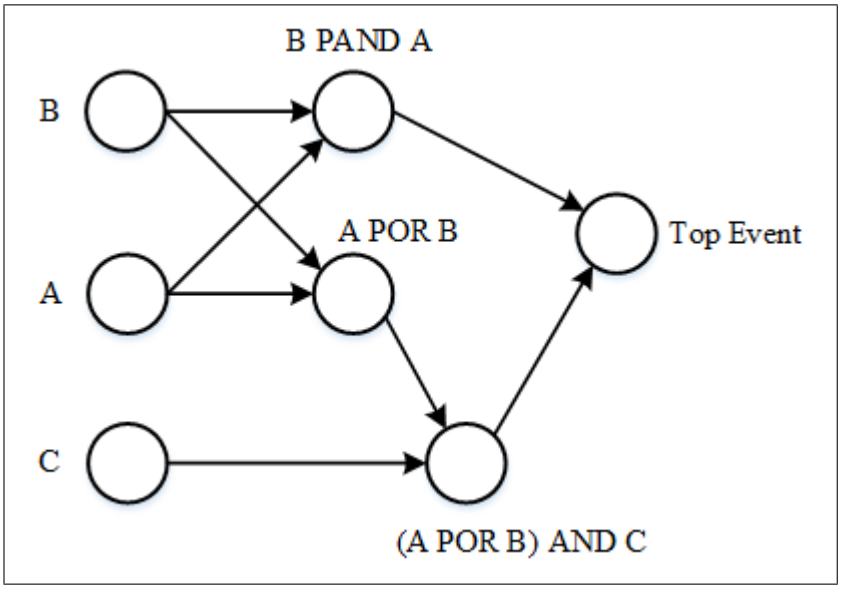


Figure 85: Bayesian Network of the TFT of Figure 77

Table 24: Prior probability values for the basic events

Basic Events	**Probabilities**					
	State 0	**State 1**	**State 2**	**State 3**	**State 4**	**State 5**
A	0.4966	0.1306	0.1136	0.0988	0.0858	0.0746
B	0.6065	0.0952	0.0861	0.0779	0.0705	0.0638
C	0.4066	0.1647	0.1376	0.1150	0.0959	0.0802

4.4.4 Observation Based Analysis and Importance Measure using BN based Method

During the probabilistic evaluation of the system unreliability, the statuses of the system components, i.e., which state the components are in, are not taken into account. That means the system reliability is evaluated solely based on the lifetime distribution of the components, and this is the only way available during the design time because at that time there is no way to know whether a particular component is failed or working, and if failed when it failed. However, when the system is implemented and employed to perform its operation, then in the operational mode, it is possible to observe the status of the components. It may not be possible to observe the status of all the components because the system may consist of thousands of components or due to the nature of the operational environment; however, we cannot rule out the possibility of observing the status of at least some of the components. So if the observation about the states of the system components can be taken into account then it is possible to investigate their effects on the system unreliability.

In the proposed BN based method, we can put observation on any of the nodes. In general, in Bayesian Networks, the probability of a child node X_i is conditionally dependent on its parent nodes, and defined by conditional probability distribution as: $Pr\{X_i|Parents(X_i)\}$. If we are not certain about the status of the parent nodes then we cannot readily say anything about the probability of the child node with certainty. To obtain the probability of the child node, we have to take all possible states of the parent nodes into account. However, certainty about the status of all or some of the parent nodes can make the status of the child node certain. For example, let us consider the BN of two input OR gate in Figure 86 where all nodes are Boolean (can be either in true of false state).

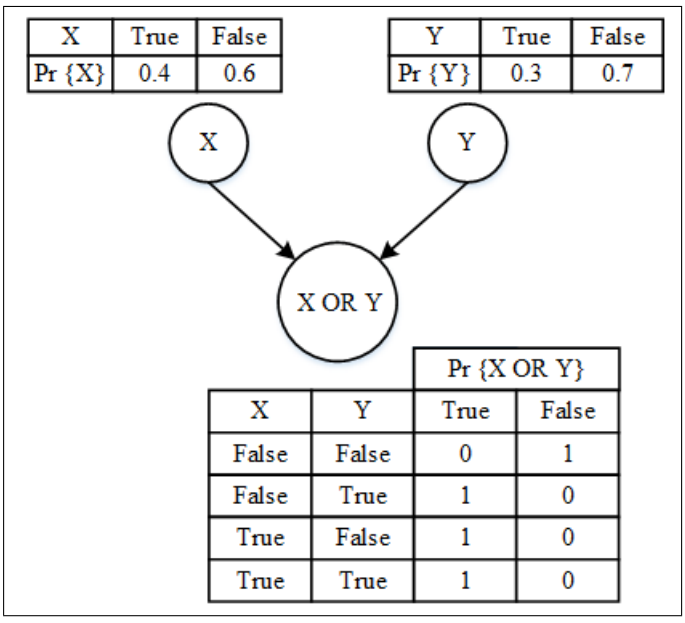


Figure 86: BN of two input Boolean OR gate

In this BN, X and Y are the parent nodes for node $X\ OR\ Y$. Now, if we want to know the probability of the node $X\ OR\ Y$ being true, $Pr\{X\ OR\ Y = True\}$, without any observation about the parent nodes then we have to do the following:

$$Pr\{X\ OR\ Y = True\} = Pr\{X\ OR\ Y = True \mid X = False, Y = False\} \times Pr\{X = False\} \times Pr\{Y = False\} + Pr\{X\ OR\ Y = True \mid X = False, Y = True\} \times Pr\{X = False\} \times Pr\{Y = True\} + Pr\{X\ OR\ Y = True \mid X = True, Y = False\} \times Pr\{X = True\} \times Pr\{Y = False\} + Pr\{X\ OR\ Y = True \mid X = True, Y = True\} \times Pr\{X = True\} \times Pr\{Y = True\}$$

$$Pr\{X\ OR\ Y = True\} = 0 \times 0.6 \times 0.7 + 1 \times 0.6 \times 0.3 + 1 \times 0.4 \times 0.7 + 1 \times 0.4 \times 0.3 = 0.58$$

The above procedure takes into account all possible states of the parent nodes (X, Y) to obtain the probability of the child node. Now, if we know the state of both the parent nodes, e.g., both the parents are in False state, then we can readily say that the probability of the node $X\ OR\ Y$ being in the true state, $Pr\{X\ OR\ Y = True \mid X = False, Y = False\}$, is 0. Now, if we can observe the state of one of the parent nodes then it will have effect on the probability of the child node. For example, if we observe the node X to be in the False state, $Pr\{X = False\} = 1$, then $Pr\{X\ OR\ Y = True\}$ would be calculated as:

$$Pr\{X\ OR\ Y = True \mid X = False\} = Pr\{X\ OR\ Y = True \mid X = False, Y = False\} \times Pr\{X = False\} \times Pr\{Y = False\} + Pr\{X\ OR\ Y = True \mid X = False, Y = True\} \times Pr\{X = False\} \times Pr\{Y = True\}$$

$$Pr\{X\ OR\ Y = True \mid X = False\} = 0 \times 1 \times 0.7 + 1 \times 1 \times 0.3 = 0.3$$

So in a two input OR gate if we know that one input is False then the probability of the OR gate becomes completely dependent on the probability of the remaining input. In the BN model of Pandora TFT, the nodes representing the basic events can be in more than two states which depends on the number of intervals used to divide the mission time. Now, if we can observe the state of any of the basic events, i.e., when the events occur then we can put that observation in the node representing that basic event, and subsequently run query on the node representing top event to see the effect. If we recall the example BN from the previous section, then we can see that without any observation about the state of the components the estimated system unreliability is 0.305. For illustration, we can try running the queries by putting some experimental observations on the root nodes. For example, if we observe the node B to be in State 4 then the system unreliability changes to 0.278, and if we observe node B to be in State 1 then the unreliability value becomes 0.373. From these results, we can conclude that the later the event B occurs the higher the reliability of the system. And it is the opposite for event A in the sense that if we observe the event A to be in State 4 then system unreliability is 0.656 and if A is observed to be

in State 1 then system unreliability becomes 0.536.

We can take advantage of this observation based analysis to determine the contribution of the components in the top events. Similar to the approach shown in section 4.3.5 for the PN based method, the importance of a component in the BN based approach is calculated by taking the difference between the top event probability and the top event probability with the component fully available. To make a component fully available, we have to observe the component to be in the State 0, meaning the component has not failed during the selected mission time. So the importance measure of a basic event BE_i is defined as:

$$IM(BE_i) = Pr\{TE\} - Pr\{TE \mid BE_i = State\ 0\} \tag{24}$$

The importance measure calculation process for the root nodes (basic events) of the BN of Figure 85 is shown in Table 25.

Table 25: Importance ranking for the basic events of the BN of Figure 85

Basic Events (BE_i)	**Pr{TE}**	**Pr{TE|BE_i = State 0}**	**IM (BE_i)**	**Rank**
A	0.305	0	0.305	1
B		0.299	0.006	3
C		0.076	0.229	2

So far, we have performed predictive reasoning on the Bayesian Network to obtain system unreliability and to determine the criticality of components. This is done by following the direction of the BN arcs from the root nodes towards the leaf nodes. In this process, failure probability data of the root nodes are used to obtain the probability of system failure, i.e., data about causes are used to obtain new belief about the effect. In the importance measure process, observation on the root nodes (causes) are provided to obtain their effects on the leaf node (effects). That means it is possible to observe the change in effects due to the change in the causes.

Using this facility of observing the status of a node, we can also perform diagnostic analysis on the BN, i.e., reasoning from effects to causes. To facilitate the diagnostic analysis, the Bayes theorem shown in equation (15) will be used. For example, consider the BN of the two input OR gate in Figure 86 where X and Y are two root nodes and the node $X\ OR\ Y$ is the leaf node. For diagnostic analysis, let us observe the leaf node being true, i.e., $Pr\{X\ OR\ Y = True\}$. Based on this observation on the leaf node we can determine the change in the root nodes. Let us say, we now want to know the probability of the node X being true. To find this what we essentially need to know is the probability of X being true given that $X\ OR\ Y$ is true, i.e., $Pr\{X = True \mid X\ OR\ Y = True\}$. We can

find this value by using the Bayes theorem as follows:

$$Pr\{X = True \mid X\ OR\ Y = True\} = \frac{Pr\{X\ OR\ Y = True \mid X = True\} \times\ Pr\{X = True\}}{Pr\{X\ OR\ Y = True\}}$$
$$= \frac{1 \times\ 0.4}{0.58} = 0.6897$$

Similarly, in reliability engineering, if the analysts have the evidence that the system has failed then based on this evidence the analysts' belief about the failure probability of the components can be updated. That means we now have to put an observation on the leaf node of the BN and work backwards (in the opposite direction of the BN arcs) towards the root nodes to update probability information of the root nodes. For illustration the node *Top Event* of the BN in figure 85 is observed to be true and the probability values of the root nodes as shown in Table 24 are updated to obtain the values shown in Table 26.

Table 26: Posterior probabilities of root nodes of the BN in Figure 85

Basic Events	**Probabilities**					
	State 0	**State 1**	**State 2**	**State 3**	**State 4**	**State 5**
A	0	0.2297	0.2162	0.2010	0.1846	0.1685
B	0.5935	0.1162	0.0950	0.0779	0.0642	0.0532
C	0.1006	0.2496	0.2086	0.1743	0.1453	0.1216

4.5 Fuzzy set theory based methodology

The analytical solutions used in HiP-HOPS and the PN and the BN based methods proposed in the last two sections for quantitative evaluation of Pandora temporal fault trees rely on the exact failure rates of system components. If the exact values of the failure rates of the basic events are known then HiP-HOPS can evaluate the top event probability. However, if the exact failure rates of components are not available then the quantitative analysis of failure behaviour of system needs to be discontinued.

It is not unusual to have components with missing failure rates or probabilities. For many complex systems, it is often very difficult to estimate the precise failure rate of components from past operational history due to lack of knowledge, limited statistical data, and changes in operating environments of the systems (Ferdous *et al.*, 2009). This situation is especially relevant in the early design phases, when the designer may have to consider components which have never used before or undetermined components which have no available failure data, and thus the exact failure rate could not possibly be known. In such situations, human judgement in linguistic terms, e.g., '*very low, low, high*' etc. can be used to determine the uncertain failure rates or probability of components, thus

allowing the quantitative analysis under uncertainty. In the literature, fuzzy sets are found as a widely used method to be used in association with classical fault trees to obtain uncertain data about the system components and allow quantitative analysis of system behaviour by means of FTA.

The main idea behind a fuzzy set theory based methodology is to use a fuzzy representation of the failure data instead of single value, and then evaluate the top event as a range of possible values. In this way, some valuable quantitative information about the dependability of a system can be obtained even if the exact data about the system components are not available. To be able to use the fuzzy representation of the failure data we have to define the fuzzy operators for the temporal fault tree gates. After the temporal fault tree is obtained using qualitative analysis, the following steps are required to be able to use the fuzzy representation of failure data in the quantitative analysis:

- Obtain fuzzy possibility of component failure data.
- Use fuzzy possibility values and the fuzzy operators of the TFT gates to obtain fuzzy top event possibility and importance measures.
- Determine the crisp top event probability from the top event possibility.

The above steps are described one by one in the next subsections.

4.5.1 Process of obtaining fuzzy failure possibility data for system components

Before we can use the fuzzy set theory based methodology, we need to decide what form of the fuzzy representation of the numbers we are going to use. After that, we have to obtain the fuzzy failure possibility of components in the prespecified format. There are different methods available to obtain fuzzy numbers such as $3\sigma\sigma$ expression or expert knowledge elicitation (Cai, 1996). In this thesis, we use the expert elicitation method to obtain the fuzzy failure possibility of components. However, the users have the flexibility to use any other method to obtain the data.

4.5.1.1 Fuzzy numbers to represent the possibilities of Basic events

Fuzzy set theory has been developed to deal with imprecise, vague or partially true information (Zadeh, 1965). A fuzzy number A can be thought of as a set of real numbers where each possible value has a weight between 0 and 1. This weight is referred to as degree of membership defined by a membership function. Let us consider a function $\mu_A(x) : \mathbb{R} \rightarrow [0,1]$ as:

$$\mu_A(x) = \begin{cases} \mu_A^l(x), & for\ a_1 < x < a_M, \\ \mu_A^r(x), & for\ a_M \leq x < a_2, \\ 0, & otherwise. \end{cases} \tag{25}$$

where $a_1,\ a_2,\ a_M \in \mathbb{R}$ and $\mu_A^l(x),\ \mu_A^r(x) : \mathbb{R} \to [0,1]$.
The above function highlights that $\mu_A(x)$ has a left region $\mu_A^l(x)$ and a right region $\mu_A^r(x)$ connected at maximum, $\mu_A^l(a_M) = \mu_A^r(a_M)$. Now we can define a fuzzy number $\boldsymbol{A}$ by the function in equation (25) which is called the membership function of the fuzzy number $\boldsymbol{A}$, and write

$$A \triangleq \mu_A(x) \tag{26}$$

where $\triangleq$ means *is defined as*.

Among different forms of fuzzy numbers, the triangular fuzzy number (TFN) and the trapezoidal fuzzy number (TZFN) are widely used in reliability analysis. In this thesis, we also use the triangular and the trapezoidal forms of the fuzzy number to represent the possibilities of the basic events. The triangular representation of the basic event failure possibilities can be denoted by a triplet $(a_1,\ a_2,\ a_3)$ and the corresponding membership function is written as:

$$\mu_A(x) = \begin{cases} \dfrac{x-a_1}{a_2-a_1}, & for\ a_1 < x < a_2, \\ \dfrac{a_3-x}{a_3-a_2}, & for\ a_2 \leq x < a_3, \\ 0, & otherwise. \end{cases} \tag{27}$$

A trapezoidal form of the failure possibility can be denoted by a quadruple $(a_1,\ a_2,\ a_3,\ a_4)$, and the membership function is defined as:

$$\mu_A(x) = \begin{cases} \dfrac{x-a_1}{a_2-a_1}, & for\ a_1 < x < a_2, \\ 1, & for\ a_2 \leq x \leq a_3, \\ \dfrac{a_4-x}{a_4-a_3}, & for\ a_3 < x < a_4, \\ 0, & otherwise. \end{cases} \tag{28}$$

If two fuzzy numbers $\tilde{A}_1$ and $\tilde{B}_1$ are expressed as (a_1,a_2,a_3,a_4) and (b_1,b_2,b_3,b_4) then algebraic operations between them are defined as (Tyagi, Pandey and Tyagi, 2010):
Addition:

$$\tilde{A}_1 \oplus \tilde{B}_1 = (a_1,a_2,a_3,a_4) \oplus (b_1,b_2,b_3,b_4) = (a_1+b_1, a_2+b_2, a_3+b_3, a_4+b_4) \tag{29}$$

Multiplication:

$$\tilde{A}_1 \otimes \tilde{B}_1 = (a_1,a_2,a_3,a_4) \otimes (b_1,b_2,b_3,b_4) = (a_1 \times b_1, a_2 \times b_2, a_3 \times b_3, a_4 \times b_4) \tag{30}$$

4.5.1.2 Domain expert evaluation and fuzzification of the opinion

In this evaluation step, a set of qualitative data representing the failure possibility of basic events are obtained. To obtain this, a set of experts are provided with a set of basic events from the TFT representing the failure behaviour of the system and the mission time

t. The experts will then subjectively evaluate the failure possibility of the components after the specified mission time. An expert is a person who is familiar with the system under consideration, has knowledge about the working environment of the system, and has considerable training and knowledge of the system operation. The experts can be selected from different fields like design, operation, maintenance, and management of the system.

Experts are human beings, therefore, they may have different levels of expertise, working experience and obviously their background may vary widely. The experts make decision about different basic events based on their experiences and their knowledge about the system. As a result, the opinions obtained from different experts are subjective due to the varying perceptions of the experts about the system. In a real world scenario, the opinion of an expert with higher experience and expertise should be given higher priority over the opinion of the expert with relatively low expertise and experience. To facilitate this, a weighting factor is used to define the relative quality of the opinion of the experts.

Table 27: Weighting scores for different experts (Rajakarunakaran *et al.*, 2015)

Constitution	**Classification**	**Score**
Professional Position	Professor, GM/DGM, Chief Engineer, Director	5
	Assistant Professor, Manager, Factory Inspector	4
	Engineer, Supervisors	3
	Foreman, Technician, Graduate apprentice	2
	Operator	1
Professional Experience (years)	≥ 20	5
	15 to 19	4
	10 to 14	3
	5 to 9	2
	< 5	1
Educational or Technical qualification	Ph.D or M.Tech.	5
	M.Sc or B.Tech.	4
	Diploma or B.Sc.	3
	ITI	2
	Secondary school	1

Due to the complexity of the systems and the vagueness of events, the experts cannot provide the exact numerical values regarding the failure possibility of components; instead they give their opinion in linguistic terms. The values of linguistic variables are words or sentences in natural languages and they play an important role in dealing with situations which are too complex or vague in nature, i.e., very difficult to describe using conventional quantitative expressions. For instance, we can consider "failure possibility of component" as a linguistic variable consisting of fuzzy sets like *very low, low, fairly low, medium, fairly high, high, very high* as shown in the example Figure 87.

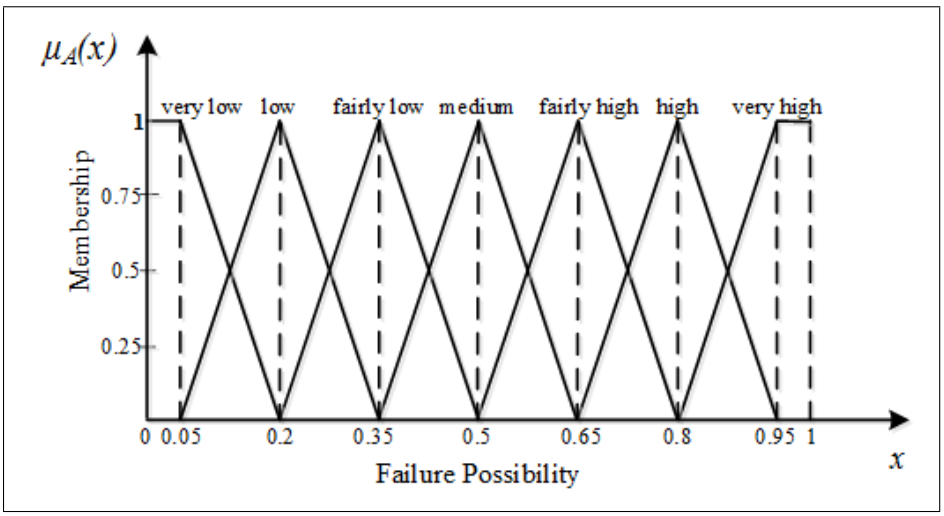


Figure 87: Fuzzy numbers representing linguistic variables

$$
\mu_{very\ low}(x) = \begin{cases} 1, & \text{if } 0 \le x \le 0.05, \\ \frac{0.2-x}{0.15}, & \text{if } 0.05 < x < 0.2, \\ 0, & \text{if } x \ge 0.2. \end{cases}
$$

$$
\mu_{low}(x) = \begin{cases} \frac{x-0.05}{0.15}, & \text{if } 0.05 < x \le 0.2, \\ \frac{0.35-x}{0.15}, & \text{if } 0.2 < x < 0.35, \\ 0, & \text{otherwise.} \end{cases}
$$

$$
\mu_{fairly\ low}(x) = \begin{cases} \frac{x-0.2}{0.15}, & \text{if } 0.2 < x \le 0.35, \\ \frac{0.5-x}{0.15}, & \text{if } 0.35 < x < 0.5, \\ 0, & \text{otherwise.} \end{cases}
$$

$$
\mu_{medium}(x) = \begin{cases} \frac{x-0.35}{0.15}, & \text{if } 0.35 < x \le 0.5, \\ \frac{0.65-x}{0.15}, & \text{if } 0.5 < x < 0.65, \\ 0, & \text{otherwise.} \end{cases} \tag{31}
$$

$$
\mu_{fairly\ high}(x) = \begin{cases} \frac{x-0.5}{0.15}, & \text{if } 0.5 < x \le 0.65, \\ \frac{0.8-x}{0.15}, & \text{if } 0.65 < x < 0.8, \\ 0, & \text{otherwise.} \end{cases}
$$

$$
\mu_{high}(x) = \begin{cases} \frac{x-0.65}{0.15}, & \text{if } 0.65 < x \le 0.80, \\ \frac{0.95-x}{0.15}, & \text{if } 0.8 < x < 0.95, \\ 0, & \text{otherwise.} \end{cases}
$$

$$
\mu_{very\ high}(x) = \begin{cases} 0, & \text{if } x \le 0.80, \\ \frac{x-0.8}{0.15}, & \text{if } 0.8 < x < 0.95, \\ 1, & \text{if } 0.95 \le x \le 1. \end{cases}
$$

Once an expert provides his/her opinion about the failure possibility of an event in linguistic terms, then this must be mapped to corresponding quantitative data in the form of a membership function of fuzzy numbers. As mentioned earlier, the membership functions could be of triangular or trapezoidal form. For example, Figure 87 shows membership functions of the linguistic variables in the triangular form and their corresponding equations are shown in equation (31). The system analysts have to define the failure possibility distribution (values for membership functions) for different linguistic variables based on the nature of the system they want to analyse.

However, this is a subjective task and results may vary from analyst to analyst. Ross (2004a,b) described six different methods — intuition, inductive reasoning, inference, genetic algorithms, neural networks, and rank ordering to form membership functions of fuzzy sets. Analysts can choose any of the above mentioned methods to define the values for membership functions of the failure possibility of the basic events.

4.5.1.3 Aggregation of the opinion of the experts

In the domain expert evaluation process, failure possibility data for each basic event is obtained from a set of M different experts. Since each expert may have different view about an event, therefore their opinion about the event may be different. In order to achieve an agreement among the conflicted views of the experts, their opinion should be aggregated into a single opinion. The aggregation could done by simply taking the arithmetic average of different opinions, but it will give all the experts equal weight and thus overlooks the knowledge, expertise and experience of the experts. On the other hand, if we take the weighted average of the opinions to obtain a single opinion then the opinion of the experienced experts (with higher score) would dominate the result, and consequently the opinion of the less experienced experts (with a low score) will not be properly reflected. To ensure that the expertise and experience of experts are taken into account and at the same time the opinions of the less experienced experts are properly accommodated in the aggregated opinion. The aggregation process is done in six different steps and the steps are described below:

Step 1: Similarity Measures

In this step, a matrix known as the *similarity matrix* (SM) is obtained in the following form by calculating similarities between the opinions of different experts.

$$SM = \begin{bmatrix} 1 & s_{12} & s_{13} & \dots & s_{1M} \\ s_{21} & 1 & s_{23} & \dots & s_{2M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{M1} & s_{M2} & s_{M3} & \dots & 1 \end{bmatrix}$$

To determine similarity between the opinions of two experts, we use the concept described by Hsu and Chen (1996). Hsu and Chen (1996) used the following equation to obtain the

similarity between two fuzzy sets.

$$S(\tilde{A}_i, \tilde{A}_j) = \frac{\int_x \left(min\{\mu_{\tilde{A}_i}(x), \mu_{\tilde{A}_j}(x)\}\right) dx}{\int_x \left(max\{\mu_{\tilde{A}_i}(x), \mu_{\tilde{A}_j}(x)\}\right) dx} \tag{32}$$

where $S(\tilde{A}_i, \tilde{A}_j)$ is known as the similarity measure function introduced by Zwick, Carlstein and Budescu (1987) and $\tilde{A}_i$, and $\tilde{A}_j$ are the opinion of expert i and j respectively. The formula in equation (32) calculates the ratio of the consistent area (overlapped area) to the whole area for any forms of fuzzy set. The overlapping between two expert opinions (in triangular form) is shown in Figure 88.

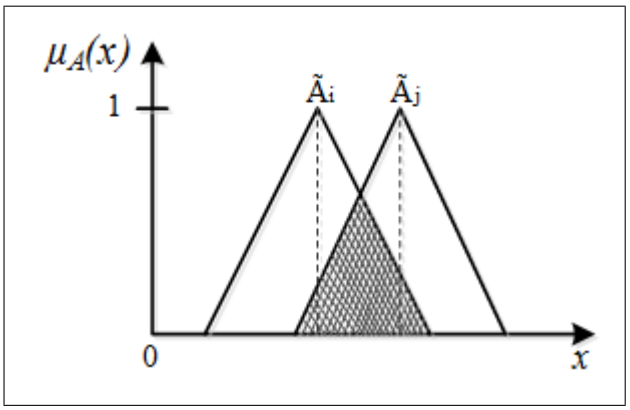


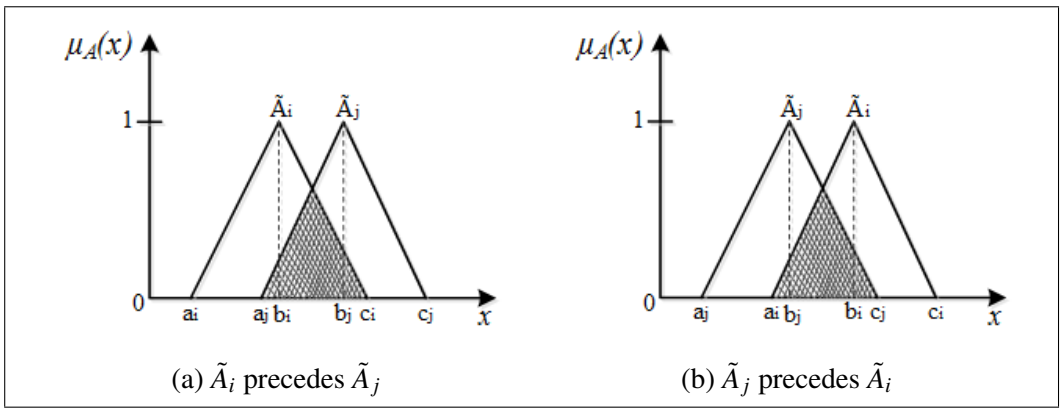
Figure 88: Overlapping between two triangular fuzzy sets

As seen in the matrix *SM*, the diagonal entries are 1, because these entries represent $S(\tilde{A}_i, \tilde{A}_i)$, i.e., similarity of one expert opinion with itself. If two opinions do not overlap at all, then the similarity between them would be 0. As mentioned earlier, in this thesis, we will use only the triangular and trapezoidal forms of the fuzzy set, therefore we have to derive specific formulae for different cases of triangular and trapezoidal fuzzy sets with the help of the formula in equation (32).

Let us first derive the formula for similarity measures for two triangular fuzzy sets. If two experts Ex_i and Ex_j provide their opinion as triangular fuzzy numbers $\tilde{A}_i$ and $\tilde{A}_j$ respectively, then four possible scenarios can occur:

1. The two sets completely overlap each other.
2. Two sets do not overlap at all.
3. Two sets partially overlap where $\tilde{A}_i$ starts before $\tilde{A}_j$ (see Figure 89a).
4. Two sets partially overlap where $\tilde{A}_j$ starts before $\tilde{A}_i$ (see Figure 89b)

In the first scenario, both the sets (opinions) are same, i.e., similarity between them is 1. In the second case, as the sets do not overlap at all, there is no similarity between them. In the other two cases, the opinions partially overlap and the overlapped area is another triangular fuzzy set.



(a) $\tilde{A}_i$ precedes $\tilde{A}_j$ (b) $\tilde{A}_j$ precedes $\tilde{A}_i$

Figure 89: Overlapping between two opinions in triangular form

Let us consider the opinion of two experts Ex_i and Ex_j as $\tilde{A}_i = \{a_i, b_i, c_i\}$ and $\tilde{A}_j = \{a_j, b_j, c_j\}$ then the similarity between the opinions can be calculated as follows:

1. If $c_i \leq a_j$ *or* $c_j \leq a_i$ then

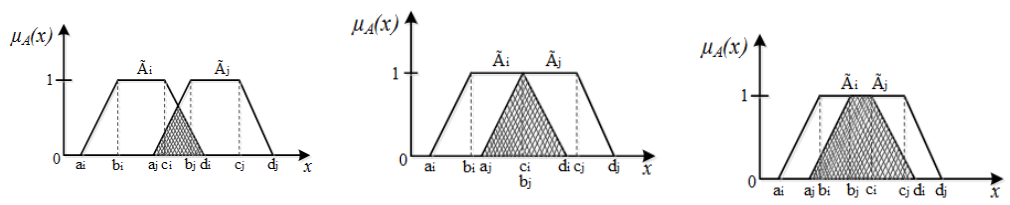
$$S(\tilde{A}_i, \tilde{A}_j) = 0. \tag{33}$$

2. If $a_i \leq a_j$ *and* $c_i > a_j$ then

$$S(\tilde{A}_i, \tilde{A}_j) = \frac{(c_i - a_j)^2}{a_i(b_i - b_j - c_i) + a_j(a_i + b_i - b_j - c_j) + b_i(-c_i - c_j) + c_j(b_j + c_i) + b_j c_i} \tag{34}$$

3. If $a_j \leq a_i$ *and* $c_j > a_i$ then

$$S(\tilde{A}_i, \tilde{A}_j) = \frac{(c_j - a_i)^2}{a_j(b_j - b_i - c_j) + a_i(a_j + b_j - b_i - c_i) + b_j(-c_j - c_i) + c_i(b_i + c_j) + b_i c_j} \tag{35}$$



(a) Triangular overlap (height < 1) (b) Triangular overlap (height = 1) (c) Trapezoidal overlap

Figure 90: Overlapping between two opinions in trapezoidal form where $\tilde{A}_i$ precedes $\tilde{A}_j$

If we use trapezoidal fuzzy sets, then four scenarios can occur like the case of triangular fuzzy sets. However, in the last two scenarios, depending on how we defined the membership functions for the linguistic variables, the overlapped area could be either a triangular or a trapezoidal fuzzy set. Figure 90 shows three possible overlapping scenarios between the opinions of two experts. In all these scenarios, in terms of values, the opinion of expert Ex_i precedes the opinion of expert Ex_j. Similar to Figure 90, three

other scenarios are possible where the opinion of expert Ex_j would precede the opinion of expert Ex_i.

Now, if we consider the opinion of two experts Ex_i and Ex_j as trapezoidal fuzzy sets as $\tilde{A}_i = \{a_i, b_i, c_i, d_i\}$ and $\tilde{A}_j = \{a_j, b_j, c_j, d_j\}$ then the similarity between the opinions can be calculated as:

1. If $d_i \leq a_j$ *or* $d_j \leq a_i$ then
$$S(\tilde{A}_i, \tilde{A}_j) = 0. \tag{36}$$

2. If $a_j < d_i$ *and* $b_j > c_i$ then
$$S(\tilde{A}_i, \tilde{A}_j) = \frac{(d_i - a_j)^2}{(b_j - c_i)(d_i - a_j) + (d_i - c_i + b_j - a_j)(-a_i + c_i - b_i + d_j + c_j - b_j)} \tag{37}$$

3. If $a_i \leq a_j$ *and* $b_j \leq c_i$ then
$$S(\tilde{A}_i, \tilde{A}_j) = \frac{d_i - a_j + c_i - b_j}{-a_i - b_i + d_j + c_j} \tag{38}$$

4. If $a_i < d_j$ *and* $b_i > c_j$ then
$$S(\tilde{A}_i, \tilde{A}_j) = \frac{(d_j - a_i)^2}{(b_i - c_j)(d_j - a_i) + (d_j - c_j + b_i - a_i)(-a_j + c_j - b_j + d_i + c_i - b_i)} \tag{39}$$

5. If $a_j \leq a_i$ *and* $b_i \leq c_j$ then
$$S(\tilde{A}_i, \tilde{A}_j) = \frac{d_j - a_i + c_j - b_i}{-a_j - b_j + d_i + c_i} \tag{40}$$

Step 2: Average agreement calculation

Once the similarity matrix is obtained, then the average agreement, $AA(Ex_i)$, for each of the experts is obtained as:

$$AA(Ex_i) = \frac{1}{M-1} \sum_{\substack{j=1 \\ j \neq i}}^{M} SM(i, j) \tag{41}$$

Step 3: Relative agreement calculation

After the average agreement for all the experts are calculated, then the relative agreement, $RAD(Ex_i)$, for all the experts is calculated as:

$$RAD(Ex_i) = \frac{AA(Ex_i)}{\sum_{i=1}^{M} AA(Ex_i)} \tag{42}$$

Step 4: Weighting factor calculation

Weighting scores for experts are defined in Table 27 based on their professional positions,

years of working experience and their educational qualifications. As a result, when we select M different experts, each of them may have different weighting score (WS). For example, if we choose a professor with a PhD degree and 20 years of work experience, then his/her weighting score would be 15 (5+5+5=15). On the other hand, the weighting score for an engineer with a MSc degree and 11 years of work experience would be 10 (3+3+4). So the weighting factor for each of the experts is calculated as:

$$WF(Ex_i) = \frac{WS(Ex_i)}{\sum_{i=1}^{M} WS(Ex_i)} \tag{43}$$

where $WS(Ex_i)$ is the weighting score of expert i and $WF(Ex_i)$ is weighting factor for expert i.

Step 5: Aggregation weight calculation

Now we have the relative agreement degree ($RAD(Ex_i)$) and weighting factor ($WF(Ex_i)$) for all of the experts calculated. To make a balance between relative agreement and weighting factor, we calculate aggregation weight as follows:

$$AW(Ex_i) = \alpha \cdot WF(Ex_i) + (1-\alpha)RAD(Ex_i) \tag{44}$$

where α ($0 \leq \alpha \leq 1$) is a relaxation factor which represents the importance of $WF(Ex_i)$ over $RAD(Ex_i)$. If α is set to zero then no importance is paid on the $WF(Ex_i)$; on the other hand if α is set to 1 then no importance is paid to $RAD(Ex_i)$. If no importance is paid to $RAD(Ex_i)$ by setting α to 1, then steps 1 to 3 are not required.

Step 6: Aggregation of opinions

This is the step where the opinions of the experts are aggregated to obtain a single opinion. The aggregation is performed using the following formula.

$$\tilde{A} = \sum_{i=1}^{M} (AW(Ex_i) \otimes \tilde{A}_i) \tag{45}$$

where $\tilde{A}$ is the aggregated opinion (a fuzzy set) and $\tilde{A}_i$ is the opinion of expert i.

4.5.2 Defuzzication and top event probability calculation

As the fuzzy possibilities of basic events are used in the quantification of the TFT, the possibilities of the minimal cut sequences as well as the top event possibility would be obtained as fuzzy numbers. In order to provide a single possibility instead of a range of possibilities, we need to map the fuzzy failure possibilities to a crisp value known as the fuzzy failure possibility score (FFPS) through defuzzification. A number of methods (Ross, 2004b; Wang, 1997) e.g., weighted average method, the centre of area method, mean max membership method, the centre of maxima method, the mean of maxima

method, centroid method, and so on are available to perform the defuzzication operation. For simplicity, in this thesis, we use the centre of area method for defuzzification. Defuzzication of a triangular fuzzy number, $\tilde{A} = (a_1, a_2, a_3)$ can be obtained using the following equation.

$$X = \frac{\int x\mu_{\tilde{A}}(x)dx}{\int \mu_{\tilde{A}}(x)dx} = \frac{\int_{a_1}^{a_2} \frac{x-a_1}{a_2-a_1} x\, dx + \int_{a_2}^{a_3} \frac{a_3-x}{a_3-a_2} x\, dx}{\int_{a_1}^{a_2} \frac{x-a_1}{a_2-a_1} dx + \int_{a_2}^{a_3} \frac{a_3-x}{a_3-a_2} dx} = \frac{1}{3}(a_1+a_2+a_3) \tag{46}$$

A trapezoidal fuzzy number, $\tilde{A} = (a_1, a_2, a_3, a_4)$ can be defuzzified as follows:

$$\begin{aligned} X &= \frac{\int_{a_1}^{a_2} \frac{x-a_1}{a_2-a_1} x\, dx + \int_{a_2}^{a_3} x dx + \int_{a_3}^{a_4} \frac{a_4-x}{a_4-a_3} x\, dx}{\int_{a_1}^{a_2} \frac{x-a_1}{a_2-a_1} dx + \int_{a_2}^{a_3} dx + \int_{a_3}^{a_4} \frac{a_4-x}{a_4-a_3} dx} \\ &= \frac{(a_4+a_3)^2 - a_4a_3 - (a_1+a_2)^2 + a_1a_2}{3(a_4+a_3-a_2-a_1)} \end{aligned} \tag{47}$$

Using equations (46) and (47) we can obtain the top event possibility as a crisp value. However, in classical FTA, the top event is quantified as a single probability value. So, we have to map the possibility value into a probability value. Onisawa (1988) has proposed a function to convert a crisp failure possibility value into a probability value. Failure probability from failure possibility can be obtained as follows.

$$FP = \begin{cases} \frac{1}{10^K}, & FFPS \neq 0, \\ 0, & FFPS = 0. \end{cases} \tag{48}$$

where FP is failure probability, $FFPS$ is fuzzy failure possibility score and $K = \left(\frac{1-FFPS}{FFPS} \right)^{\frac{1}{3}} \times 2.301$.

4.5.3 Fuzzy operators for TFT gates

Once the fuzzy failure possibilities of all the basic events are obtained, then we can use these values to quantify the top event possibility. However, we first need to define the fuzzy operators for all the TFT gates. One thing to note is that all the fuzzy operators for the TFT gates are defined for exponential distribution of failure rates and for a continuous time domain.

4.5.3.1 Fuzzy operators for the Boolean gates

Fuzzy operator for the AND gate:

The outcome of a AND gate becomes true when all the input events are true. Therefore,

the output probability of a AND gate with N inputs can be obtained as a probability of all N inputs occurring. If there are N statistically independent input events in a AND gate at time t then the probability of the output of the AND gate is (Henley and Kumamoto, 1981):

$$Pr\{E_1 \wedge E_2 \wedge E_3 \wedge \ \cdots \ \wedge E_{n-1} \wedge E_n\}(t) = \prod_{i=1}^{N} Pr\{E_i\}(t) \tag{49}$$

where $P_r\{E_i\}(t)$ is the failure probability of event E_i at time t.

Now, we do not have the probability values for the basic events, rather we have fuzzy possibility values for the basic events. So, if the failure possibility of event i is presented by a triangular fuzzy number as $P_i(t) = \{a_i(t), b_i(t), c_i(t)\}$, then the AND gate fuzzy operator for the triangular representation of the failure possibilities can be defined as:

$$\begin{aligned} P_{ANDF} &= AND_F\big(\{P_1(t), P_2(t), \cdots, P_N(t)\}\big) = \prod_{i=1}^{N} P_i(t) \\ &= \left\{ \prod_{i=1}^{N} a_i(t), \prod_{i=1}^{N} b_i(t), \prod_{i=1}^{N} c_i(t) \right\} \end{aligned} \tag{50}$$

Similarly for the trapezoidal representation of the failure possibilities, $P_i(t) = \{a_i(t), b_i(t), c_i(t), d_i(t)\}$, the AND gate fuzzy operator can be defined as:

$$\begin{aligned} P_{ANDF} &= AND_F\big\{P_1(t), P_2(t), \cdots, P_N(t)\big\} = \prod_{i=1}^{N} P_i(t) \\ &= \left\{ \prod_{i=1}^{N} a_i(t), \prod_{i=1}^{N} b_i(t), \prod_{i=1}^{N} c_i(t), \prod_{i=1}^{N} d_i(t) \right\} \end{aligned} \tag{51}$$

Fuzzy operator for the OR gate:

In HiP-HOPS, the OR gates are usually quantified using the Esary-Proschan formula (Esary and Proschan, 1963). According to the formula, if there are N input events in an OR gate at time t then the probability of the output of the OR gate is:

$$Pr\{E_1 \vee E_2 \vee E_3 \vee \cdots \vee E_{n-1} \vee E_n\}(t) = 1 - \prod_{i=1}^{N} \big(1 - P_r\{E_i\}(t)\big) \tag{52}$$

In the absence of the failure probability, if the failure possibility of the event E_i is presented by a triangular fuzzy number as $P_i(t) = \{a_i(t), b_i(t), c_i(t)\}$, then the OR gate fuzzy operator for the triangular representation of the failure possibilities can be defined as:

$$\begin{aligned} P_{ORF} &= OR_F\big\{P_1(t), P_2(t), \cdots, P_N(t)\big\} = 1 - \prod_{i=1}^{N} \big(1 - P_i(t)\big) \\ &= \left\{ 1 - \prod_{i=1}^{N} \big(1 - a_i(t)\big), 1 - \prod_{i=1}^{N} \big(1 - b_i(t)\big), 1 - \prod_{i=1}^{N} \big(1 - c_i(t)\big) \right\} \end{aligned} \tag{53}$$

For trapezoidal representation of the failure possibilities, the OR gate fuzzy operator is defined as:

$$P_{ORF} = OR_F\{P_1(t), P_2(t), \cdots, P_N(t)\} = 1 - \prod_{i=1}^{N}\big(1 - P_i(t)\big)$$
$$= \left\{1 - \prod_{i=1}^{N}\big(1 - a_i(t)\big), 1 - \prod_{i=1}^{N}\big(1 - b_i(t)\big), 1 - \prod_{i=1}^{N}\big(1 - c_i(t)\big), 1 - \prod_{i=1}^{N}\big(1 - d_i(t)\big)\right\} \tag{54}$$

4.5.3.2 Fuzzy operator for the Temporal gates

As seen from equations (9) and (10) that to obtain probability of the PAND and the POR gate we need to know the failure rate of components. However, at present we have the failure possibility of components in the fuzzy form. So, we have to obtain the fuzzy failure rate of components from the fuzzy failure possibility of the components. This can be done in two steps. In the first step, the fuzzy possibility will be converted to fuzzy probability (FP) using equation (48).

In the second step, the fuzzy failure rate can be obtained using the following equation.

$$\lambda = \frac{-\ln(1 - FP)}{t} \tag{55}$$

where λ is the failure rate and t is the mission time.

Fuzzy operator for the PAND gate:
In a minimal cut sequence (MCSQ), if there are N statistically independent input events in a PAND gate and they occur sequentially, i.e., event 1 occurs first, then event 2,$\cdots$, then event N–1, and finally event N, then the fuzzy probability of that PAND gate can be defined using equation (9) as:

$$P_{PANDF} = PAND_F\{\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_n\}(t) = \prod_{i=1}^{N}\lambda_i \sum_{k=0}^{N}\left[\frac{e^{(u_k t)}}{\prod_{\substack{j=0 \\ j\neq k}}^{N}(u_k - u_j)}\right] \tag{56}$$

where λ_i is the fuzzy representation of failure rate, $u_0 = 0$ and $u_k = -\sum_{j=1}^{k}\lambda_j \; for \; k > 0$. If the failure rate of an event E_i is represented by a triangular fuzzy number as (l_i, m_i, n_i),then the fuzzy probability of the outcome of the PAND gate can be defined as:

$$P_{PANDF} = \left\{\prod_{i=1}^{N} l_i \sum_{k=0}^{N}\left[\frac{e^{(u_k t)}}{\prod_{\substack{j=0 \\ j\neq k}}^{N}(u_k - u_j)}\right], \prod_{i=1}^{N} m_i \sum_{k=0}^{N}\left[\frac{e^{(u_k t)}}{\prod_{\substack{j=0 \\ j\neq k}}^{N}(u_k - u_j)}\right], \right.$$
$$\left. \prod_{i=1}^{N} n_i \sum_{k=0}^{N}\left[\frac{e^{(u_k t)}}{\prod_{\substack{j=0 \\ j\neq k}}^{N}(u_k - u_j)}\right]\right\} \tag{57}$$

Similarly for the trapezoidal representation of the failure rates, $\lambda_i = (l_i, m_i, n_i, o_i)$, the

fuzzy probability of the outcome of the PAND gate can be defined as:

$$P_{PANDF} = \left\{ \prod_{i=1}^{N} l_i \sum_{k=0}^{N} \left[\frac{e^{(u_k t)}}{\prod_{\substack{j=0 \\ j\neq k}}^{N} (u_k - u_j)} \right], \prod_{i=1}^{N} m_i \sum_{k=0}^{N} \left[\frac{e^{(u_k t)}}{\prod_{\substack{j=0 \\ j\neq k}}^{N} (u_k - u_j)} \right], \right.$$
$$\left. \prod_{i=1}^{N} n_i \sum_{k=0}^{N} \left[\frac{e^{(u_k t)}}{\prod_{\substack{j=0 \\ j\neq k}}^{N} (u_k - u_j)} \right], \prod_{i=1}^{N} o_i \sum_{k=0}^{N} \left[\frac{e^{(u_k t)}}{\prod_{\substack{j=0 \\ j\neq k}}^{N} (u_k - u_j)} \right] \right\} \tag{58}$$

If there are 2 input events in a PAND gate, then according to Merle *et al.* (2010), equation (56) reduces to:

$$P_{PANDF} = \frac{\lambda_2}{\lambda_1 + \lambda_2} e^{-(\lambda_1+\lambda_2)t} - e^{-\lambda_2 t} + \frac{\lambda_1}{\lambda_1 + \lambda_2} \tag{59}$$

In this case, if the failure rates of the two events are represented as $\lambda_1 = (l_1, m_1, n_1, o_1)$ and $\lambda_2 = (l_2, m_2, n_2, o_2)$ then the fuzzy probability of the PAND gate can be written as:

$$P_{PANDF} = \left\{ \begin{array}{l} \dfrac{l_2}{l_1 + l_2} e^{-(l_1+l_2)t} - e^{-l_2 t} + \dfrac{l_1}{l_1 + l_2}, \\ \dfrac{m_2}{m_1 + m_2} e^{-(m_1+m_2)t} - e^{-m_2 t} + \dfrac{m_1}{m_1 + m_2}, \\ \dfrac{n_2}{n_1 + n_2} e^{-(n_1+n_2)t} - e^{-n_2 t} + \dfrac{n_1}{n_1 + n_2}, \\ \dfrac{o_2}{o_1 + o_2} e^{-(o_1+o_2)t} - e^{-o_2 t} + \dfrac{o_1}{o_1 + o_2} \end{array} \right\} \tag{60}$$

Fuzzy operator for the POR gate:

For any minimal cut sequence of N statistically independent events in a POR gate with the expression $E_1 \wr E_2 \wr E_3 \wr \cdots \wr E_{N-1} \wr E_N$, and the fuzzy failure rates $\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_{N-1}, \lambda_N$ respectively, then the fuzzy probability of the POR gate can be defined using equation (10) as:

$$P_{PORF} = POR_F\{\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_n\}(t) = \frac{\lambda_1 \left(1 - \left(e^{-(\sum_{i=1}^{N} \lambda_i)t}\right)\right)}{\sum_{i=1}^{N} \lambda_i} \tag{61}$$

If the failure rate of event E_i is represented by a triangular fuzzy number as $\lambda_i = (l_i, m_i, n_i)$, then the fuzzy probability of the outcome of the POR gate can be defined as:

$$P_{PORF} = \left\{ \frac{l_1 \left(1 - \left(e^{-(\sum_{i=1}^{N} l_i)t}\right)\right)}{\sum_{i=1}^{N} l_i}, \frac{m_1 \left(1 - \left(e^{-(\sum_{i=1}^{N} m_i)t}\right)\right)}{\sum_{i=1}^{N} m_i}, \right.$$
$$\left. \frac{n_1 \left(1 - \left(e^{-(\sum_{i=1}^{N} n_i)t}\right)\right)}{\sum_{i=1}^{N} n_i} \right\} \tag{62}$$

Similarly, for the trapezoidal representation of the failure rates, $\lambda_i = (l_i, m_i, n_i, o_i)$, of basic

events the fuzzy probability of the POR gate can be written as:

$$P_{PORF} = \left\{ \frac{l_1\left(1-\left(e^{-(\sum_{i=1}^{N} l_i)t}\right)\right)}{\sum_{i=1}^{N} l_i}, \frac{m_1\left(1-\left(e^{-(\sum_{i=1}^{N} m_i)t}\right)\right)}{\sum_{i=1}^{N} m_i}, \right. \\ \left. \frac{n_1\left(1-\left(e^{-(\sum_{i=1}^{N} n_i)t}\right)\right)}{\sum_{i=1}^{N} n_i}, \frac{o_1\left(1-\left(e^{-(\sum_{i=1}^{N} o_i)t}\right)\right)}{\sum_{i=1}^{N} o_i} \right\} \tag{63}$$

As mentioned in Section 4.2, when we consider the events as statistically independent and the mission time as continuous, then the probability of two or more events occurring exactly at the same time is zero, i.e., in the continuous time domain the probability of SAND gate with N statistically independent event is zero. As the probability of SAND gate equates to zero for the above assumption, a fuzzy operator for the SAND gate is not defined.

We can see that the fuzzy operators for the Boolean gates use the failure possibility of events as input and produce the output as a fuzzy possibilities. On the other hand, the fuzzy operators for the temporal gates use the fuzzy failure rate as input and produce the output as fuzzy probability. As the top event is represented as the logical OR of different MCSQs, we need to convert the fuzzy probability values obtained by the temporal gates into fuzzy possibility values. Failure possibility from failure probability can be obtained using equation (48) as follows:

$$FFPS = \begin{cases} \dfrac{1}{1+\left(\dfrac{K}{2.301}\right)^3}, & \text{if } FP \neq 0. \\ 0, & \text{if } FP = 0. \end{cases} \tag{64}$$

where $FFPS$ is fuzzy failure possibility score, FP is failure probability, and $K = \log_{10}\left(\dfrac{1}{FP}\right)$.

4.5.4 Importance Measures in Fuzzy Set Theory based Approach

The evaluation of the contribution of different basic events to the top event probability is very important in identifying the critical components. The importance measures used in the traditional probabilistic approaches are not applicable in the case of fuzzy set theory based approaches because in this case basic event failure data is represented as fuzzy possibilities rather than crisp probabilities. So we have to define new importance measure that are suitable for the fuzzy set theory based methodology. Different methodologies (e.g., (Guimarees and Ebecken, 1999; Suresh, Babar and Raj, 1996; Tyagi, Pandey and Kumar, 2011)) have already been proposed to quantify fuzzy importance measures.

In this thesis, we calculate the fuzzy importance of a basic event by taking the difference between the fuzzy top event possibilities with and without the presence of the basic event. Let $\tilde{P}_{T_i=1}$ be the fuzzy failure possibility of the top event with the basic event E_i

fully unavailable, i.e., fuzzy possibility of the basic event E_i is considered as $\{1, 1, 1\}$ or $\{1, 1, 1, 1\}$ in triangular or trapezoidal form. On the other hand, $\tilde{P}_{T_i=0}$ is the failure possibility of the top event when the possibility of basic event E_i is either $\{0, 0, 0\}$ or $\{0, 0, 0, 0\}$, i.e., the basic event E_i is fully available. In conventional approaches, the Birnbaum importance is obtained by taking the difference between $\tilde{P}_{T_i=1}$ and $\tilde{P}_{T_i=0}$ where $\tilde{P}_{T_i=1}$ and $\tilde{P}_{T_i=0}$ are crisp values. However, in this case, $\tilde{P}_{T_i=1}$ and $\tilde{P}_{T_i=0}$ are fuzzy numbers, hence, we need to find distance between these two numbers to find the fuzzy importance of a basic event. The distance between two fuzzy numbers can be obtained using Euclidean or Hamming distance (Deza and Deza, 2009). In this thesis, we use the Euclidean distance to obtain the distance between two fuzzy numbers. As a result, the fuzzy importance measure (FIM) for a basic event E_i is defined as:

$$FIM(E_i) = ED[\tilde{P}_{T_i=1}, \tilde{P}_{T_i=0}] \tag{65}$$

where $ED[\tilde{P}_{T_i=1}, \tilde{P}_{T_i=0}]$ is the Euclidean distance between $\tilde{P}_{T_i=1}$ and $\tilde{P}_{T_i=0}$.
If $\tilde{P}_{T_i=1} = \{{a_1}^1,\ {a_2}^1\ , {a_3}^1\}$ and $\tilde{P}_{T_i=0} = \{{a_1}^0,\ {a_2}^0\ , {a_3}^0\}$ then

$$FIM(E_i) = ED[\tilde{P}_{T_i=1}, \tilde{P}_{T_i=0}] = \sqrt{({a_1}^1 - {a_1}^0)^2 + ({a_2}^1 - {a_2}^0)^2 + ({a_3}^1 - {a_3}^0)^2} \tag{66}$$

Using the above equation we can calculate importance measure for all the basic events and rank them in accordance with their importance index. For two basic events E_i and E_j, if $FIM(E_i) > FIM(E_j)$ then the basic event E_i will have greater importance then the basic event E_j.

4.6 Case Study

This section describes the functionality of a fault tolerant fuel distribution system of a ship and demonstrates how the quantitative evaluation methods for Pandora temporal fault trees proposed in the earlier sections can be used to evaluate the reliability of such a dynamic system with uncertainty.

4.6.1 Fuel Distribution System of a Ship

The case study of the fault tolerant fuel distribution system of a ship was first used in (Edifor *et al.*, 2012) but reworked in this thesis, and shown in Figure 91.
The system consists of:

- Two identical fuel tanks to store fuel and they are connected to other system components using polythene piping.
- Three unidirectional fuel pumps to provide fuel to the engines from the tanks.

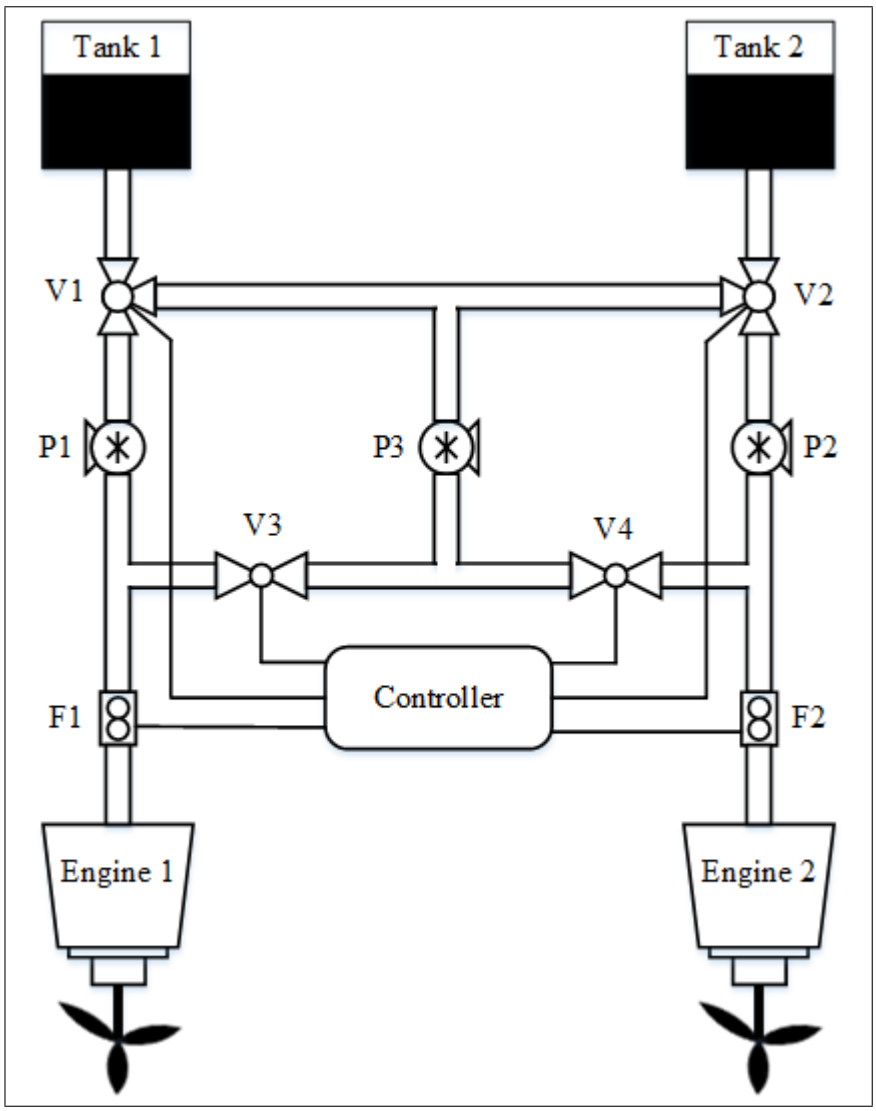


Figure 91: Fault Tolerant fuel Distribution System

- Four valves that can activate some paths or block some paths according to the requirements of the system in different situations. These valves are software controlled.
- Two flowmeters to measure the rate of fuel flow through the pipes and these measurements are used in deciding the paths to activate and deactivate to maintain the proper fuel flow to the engines.
- Two engines to provide thrust for the ship and are responsible for the manoeuvrability of the ship.
- A central controller that controls different valves to activate and deactivate different paths to maintain the proper fuel flow to the engines.

The primary functions of the system are storing fuel in the tanks during refuelling and distributing fuel to the engines during the consumption mode. In the consumption mode, the engines consume fuel to produce mechanical energy and the power of the engines are

transferred to the propellers. The propellers produce thrust for the ship and it causes the ship to move. If for some reason the engines fail to operate, then the propellers would not get the mechanical power to produce the thrust for the ship, thus the manoeuvrability of the ship will be reduced. That means the proper manoeuvrability of the ship depends on the proper functionality of the engines which is dependent on the proper fuel flows to them. The system is fault tolerant in the sense that there are two different paths for fuel flow to each of the engines and if one of the paths become unavailable then the alternative path can be used to ensure fuel flow to the engines. The Controller is the central component of the system and it is responsible for maintaining proper fuel flow to the engines. It does this by activating and deactivating different paths of fuel flow by activating the spare pump and by opening or closing different valves according to the fuel requirements.

Under functional conditions, there are two primary fuel flows: Pump 1 (P1) provides fuel to Engine 1 from Tank 1, and Pump 2 (P2) provides fuel to Engine 2 from Tank 2. Flowmeters observe the rate of fuel flow to each engine and report it back to the Controller. On detecting inadequate fuel flow to either of the engine, the Controller activates the standby Pump 3 (P3) to introduce dynamic behaviour to this system and redirects fuel flow through valves V1-V4 accordingly. For instance, if a problem is detected with the fuel flow to Engine 1, then the Controller can redirect fuel flows to Engine 1 through Pump 3 instead of Pump 1 by opening Valves 1 and 3 (V1 and V3) and activating Pump 3. In contrast, if inadequate fuel flow to Engine 2 is detected then Valves 2 and 4 (V2 and V4) will be opened instead of V1 and V3 and Pump 3 will be activated. Consequently, P3 can replace either P1 or P2, but not both. A failure of both P1 and P2 will result in at least one engine being starved of fuel; e.g., if P1 fails and P3 replaces it, then P3 will be unavailable for replacing P2 if P2 subsequently fails. This results in degraded thrust functionality for the ship and with one engine working only the manoeuvrability and speed of the ship will be reduced.

4.6.2 Qualitative Temporal Fault Tree Analysis of the System

Before quantitative analysis can proceed, the qualitative information about the failure behaviour of the system must be obtained. Pandora temporal gates can be used to model the dynamic behaviour of the above mentioned system and helps to correctly capture the sequences of events that can lead to failure. For simplicity, internal failure of the engines themselves is left out of the scope of this analysis. The Pandora TFT for the failure behaviour of Engine 1 of the fuel distribution system was constructed via model-based synthesis from Pandora descriptions of local failure logic of components.

Failure modes of the different components of the systems are abbreviated and shown in Table 28.

At the top level, the causes of omission of fuel to Engine1 and 2 can be expressed using temporal gates as follows:

Table 28: List of Basic Events for the fuel distribution system

Basic Events	Description
P1	Failure of Pump 1
P2	Failure of Pump 2
P3	Failure of Pump 3
V1	Failure of Valve 1 (e.g. blockage or stuck closed)
V2	Failure of Valve 2 (e.g. blockage or stuck closed)
V3	Failure of Valve 3 (e.g. blockage or stuck closed)
V4	Failure of Valve 4 (e.g. blockage or stuck closed)
E1	Omission of fuel to Engine 1
E2	Omission of fuel to Engine 2
S1	Failure of Flowmeter sensor 1 (e.g. sensor readings stuck high)
S2	Failure of Flowmeter sensor 2
CF	Failure of Controller

$$\texttt{O-Engine1} = (\texttt{O-Pump1} \wr \texttt{O-Pump2}) \wedge \texttt{O-Valve3}) \vee (\texttt{O-Pump2} \lhd \texttt{O-Pump1}) \vee (\texttt{O-Pump2 \& O-Pump1})$$

$$\texttt{O-Engine2} = (\texttt{O-Pump2} \wr \texttt{O-Pump1}) \wedge \texttt{O-Valve4}) \vee (\texttt{O-Pump1} \lhd \texttt{O-Pump2}) \vee (\texttt{O-Pump1 \& O-Pump2})$$

As `O-Engine1` and `O-Engine2` are caused by the same events in the opposite sequences, only the failure behaviour of Engine 1 will be considered here. Omission of fuel to Engine1 (`O-Engine1`) has three possible causes, depending on the sequence of events:

1. If there is no fuel from Pump 1 (`O-Pump1`), then Pump 3 replaces it, as long as Pump 2 has not failed first; this precondition can be represented using the POR gate. Thus in this situation, an omission of fuel can be caused by omission of fuel from both Pump 1 and Pump 3 (via Valve 3).

2. If Pump 2 fails first, then Pump 3 replaces it and will be unavailable to replace Pump 1 if it also fails. Thus sequential failure of Pump 2 and then Pump 1 will lead to an omission of fuel to Engine 1 (represented using the PAND gate).

3. If both Pump 2 and Pump 1 fail at the same time (represented with the SAND gate), then Pump 3 can only replace one of them. Behaviour in this situation is

non-deterministic (as Pump 3 may replace either Pump 1 or Pump 2, but not both), and thus as a pessimistic estimation, simultaneous failure of Pump 1 and Pump 2 is given as a cause of failure for both engines.

The expanded fault tree expressions for the failure of Engine 1 are as follows:

```
E1 = (P1∨P1≀P2≀CF≀V1)≀(P2∨P2≀P1≀CF≀V2)∧(V3∨P3∨(V1◁P1≀P2≀CF)
∨(V1&P1≀P2≀CF)∨(S1◁P1≀P2)∨(CF◁P1≀P2)∨(S1&P1≀P2)
∨(CF&P1≀P2)∨(V2◁P2≀P1≀CF)∨(V2&P2≀P1≀CF)∨(S2◁P2≀P1)
∨(CF◁P2≀P1)∨(S2&P2≀P1)∨(CF&P2≀P1))∨(P2∨P2≀P1≀CF≀V2)
◁(P1∨P1≀P2≀CF≀V1)∨(P2∨P2≀P1≀CF≀V2)&(P1∨P1≀P2≀CF≀V1)
```

After minimising the above expressions using Pandora temporal laws, the resulting minimal cut sequences to cause failure of Engine 1 are shown in Table 29.

Table 29: Minimal Cut Sequences to cause the failure of Engine 1

Minimal Cut Sequence	**Description**
$(P1 \wr P2) \wedge P3$	Failure of Pump 1 before Pump 2 (if Pump 2 fails at all) and failure of Pump 3
$(P1 \wr P2) \wedge V1$	Failure of Pump 1 before Pump 2 (if Pump 2 fails at all) and failure of Valve 1
$(P1 \wr P2) \wedge V3$	Failure of Pump 1 before Pump 2 (if Pump 2 fails at all) and failure of Valve 3
$(S1 \lhd P1) \wr P2$	Failure of Flowmeter 1 before Pump 1, as long as Pump 2 has not failed yet
$(S1 \& P1) \wr P2$	Simultaneous failure of Flowmeter 1 and Pump 1, as long as Pump 2 has not failed yet
$(CF \lhd P1) \wr P2$	Failure of Controller before Pump 1, as long as Pump 2 has not failed yet
$(CF \& P1) \wr P2$	Simultaneous failure of Controller and Pump 1, as long as Pump 2 has not failed yet
$P2 \lhd P1$	Failure of Pump 2 before Pump 1
$P1 \& P2$	Simultaneous failure of both Pump 1 and Pump 2

As mentioned earlier, it is assumed that all events are independent and the probability of two independent events occurring at the same time is effectively 0, therefore MCSQs containing SAND gates will not be considered during the quantitative analysis. Thus for this example system, minimal cut sequences: `(S1&P1)≀P2`, `(CF&P1)≀P2` and

(P1&P2) are not considered during the reliability quantification of the example system. The Pandora TFT of the failure behaviour of the fault tolerant fuel distribution system is shown in Figure 92.

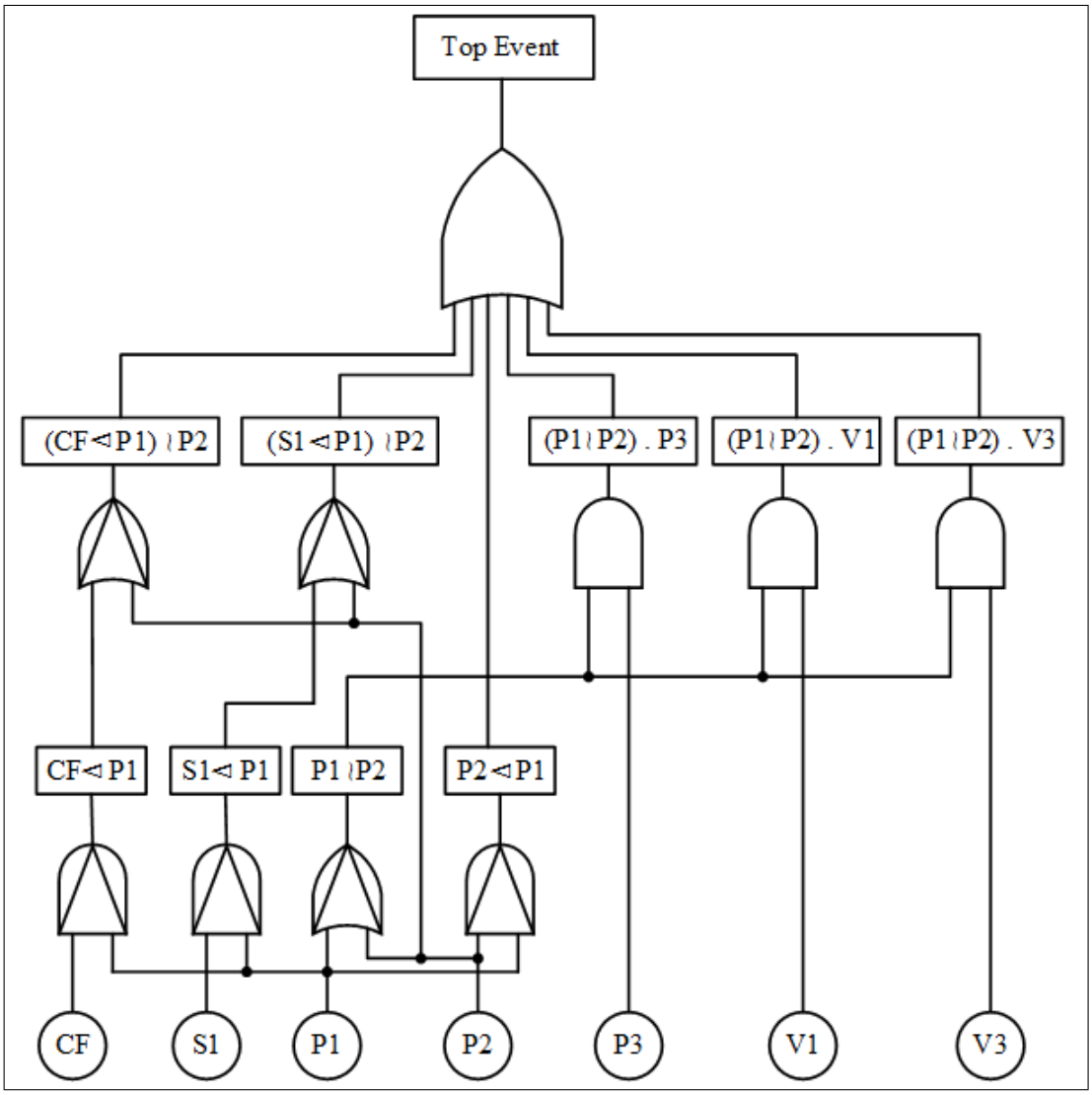


Figure 92: TFT of failure behaviour of Engine 1

4.6.3 Quantitative Analysis using the PN based Method

In order to quantify the Pandora TFT of Figure 92 to evaluate the reliability of the fuel distribution system using the PN based method, we first need to define the exponential failure rates of the components. The failure rates of the components involved in the minimal cut sequences (see Table 29) are shown in Table 30.

The Pandora TFT of the failure behaviour of the fuel distribution system is translated into Petri Net model following the procedures described in Section 4.3, and shown in Figure 93. As seen in this figure, timed transitions (white rectangles) are characterised by exponential firing rates based on the failure rate of the components they are connected with. To obtain the system unreliability from the PN model we need to simulate the model.

For simulation, we need to define the mission time and the step size. The PN model was created and simulated using the ORIS tool (Horváth *et al.*, 2012). The system unreliability was calculated for mission times ranging from 1000 hours to 20000 hours. The step size does not have any effect on the system unreliability but it controls the execution time (simulation speed). For this reason, we have tried different step sizes to observe the variations in execution time and execution time is taken as an average over 10 runs for each mission time.

Table 30: Failure rates of components of fuel distribution system

Component	**Failure rate/hour (λ)**
Pump 1	3.2 E-5
Pump 2	3.2 E-5
Pump 3	3.2 E-5
Valve1	1.0 E-5
Valve3	6.0 E-6
Flowmeter Sensor	2.5 E-6
Controller	5.0 E-7

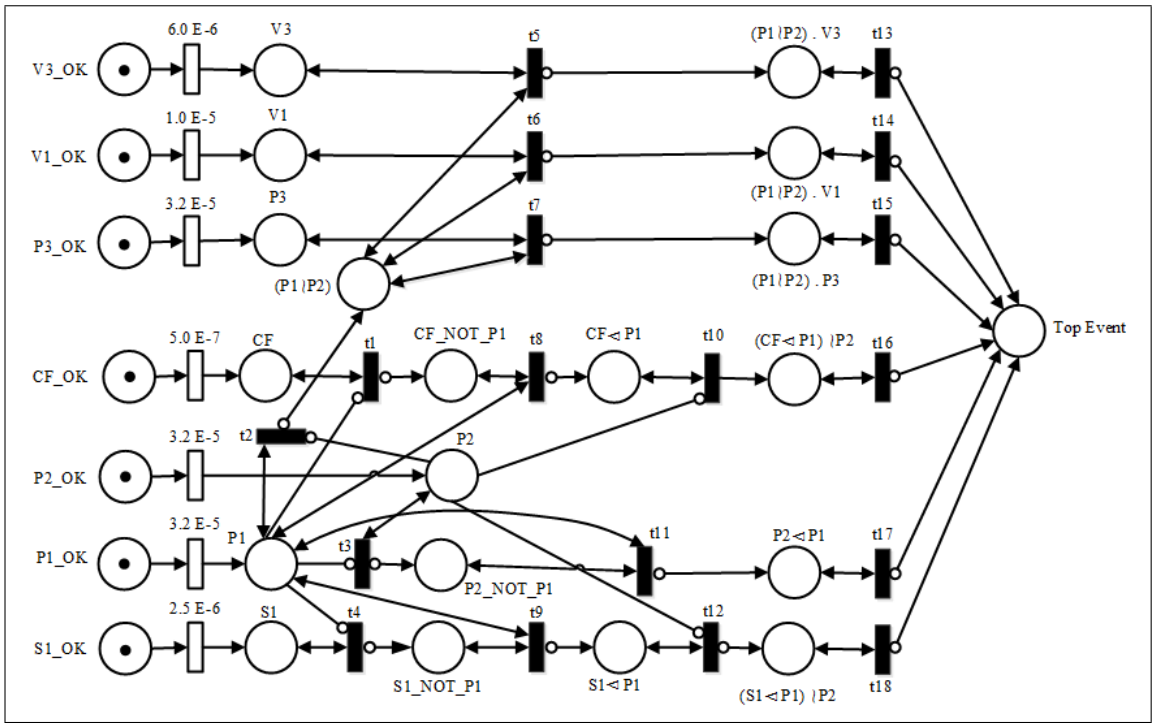


Figure 93: Petri Net model of the failure behaviour of the fuel distribution system

Table 31: Unreliability of the fuel distribution system and average execution time with different step sizes (SS)

Mission Time (Hours)	Unreliability	Average Execution Time (seconds)			
		SS=20	SS=250	SS=500	SS=1000
1000	0.0017	8.0	1.0	1.0	1.0
2000	0.0066	64.8	9.0	6.0	5.0
3000	0.0143	209.7	24.0	16.0	12.1
4000	0.0242	378.6	39.0	25.2	17.2
5000	0.0362	560.8	56.0	33.5	22.4
6000	0.0500	773.8	74.4	43.6	28.4
7000	0.0651	970.8	90.4	52.4	33.5
8000	0.0815	1146.7	104.6	60.2	37.5
9000	0.0989	1322.6	119.6	68.3	40.8
10000	0.1170	1501.0	134.9	75.5	45.0
11000	0.1359	1706.7	152.2	83.4	48.8
12000	0.1552	1858.4	166.6	89.3	52.5
13000	0.1749	2032.8	181.6	96.5	55.7
14000	0.1948	2204.4	196.7	103.4	59.3
15000	0.2149	2365.4	211.1	110.1	62.1
16000	0.2351	2520.8	223.2	116.5	65.4
17000	0.2552	2692.3	236.7	123.4	69.2
18000	0.2753	2874.4	248.7	130.1	72.3
19000	0.2952	3030.5	260.4	136.6	75.8
20000	0.3150	3182.3	270.8	143.3	79.4

The results showing the system unreliability with different mission times and the average execution time with four different step sizes are shown in Table 31. Different other values of step size were also used to calculate the average execution time and the graph in Figure 94 shows the trends in the change in the execution time with the change in the mission time and step size. As seen from the results in Table 31 and graph in Figure 94, in the PN based method, both mission time and step size have an effect on the average execution time. We can see that, as the value of mission time increases, the execution time also increases linearly. This is an expected behaviour because when the simulation runs it divides the mission time into several small intervals based on the step size and it continues accumulating the time to simulate each small interval to obtain a single value

for the whole mission time at the end of the simulation. For this reason, by increasing mission time we are adding some extra intervals, so the simulation will take some extra time for those extra intervals. For example, if an instance of the simulation divides the mission time into n intervals and each interval takes t_1 seconds on average to simulate, then the total simulation time would be $t = n \times t_1$. Now if we increase the mission time by adding m extra intervals then the simulation time will become $t = n \times t_1 + m \times t_1$, i.e., $t = (n+m) \times t_1$. This also implies that the more steps the simulation will take the more time it will take to complete. That means the smaller the step size, the higher the total simulation time, which is also seen in the graph in Figure 94. As the step size does not have any effect on the system unreliability, we can choose the maximum possible step size to keep the simulation time minimum. For this case study, as we wanted to know the system unreliability in every 1000 hours up to 20000 hours, we can choose the maximum step size as 1000.

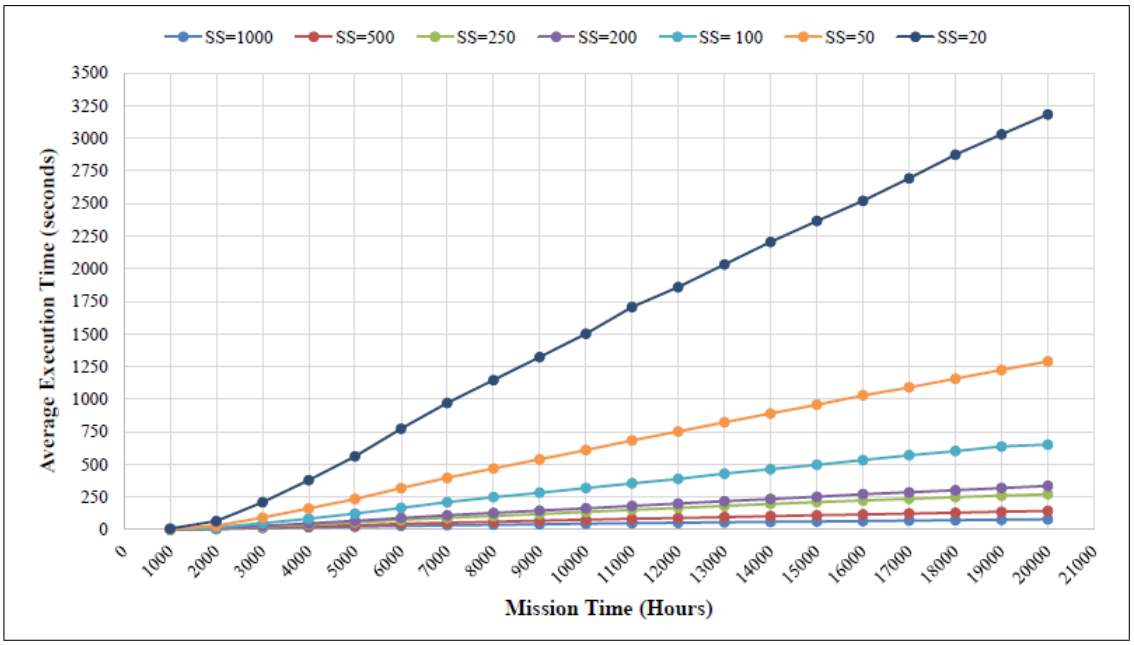


Figure 94: Changes in average execution time with the change of value of mission time and step size

One important aspect of quantitative reliability analysis of system designs is to identify the critical components so that the designers can decide where to focus their efforts on those parts of the system that require most improvement to satisfy the requirements, e.g., by applying fault tolerance strategies. The importance measures of the components of the fuel distribution system were calculated according to the procedure described in section 4.3.5, and shown in Table 32. The importance of the components were evaluated for 20000 hours of mission time.

From Table 32, it can be seen that the highest contribution to the top event is from the Pump 1 (P1) and the lowest contribution is from the Controller (CF). So, for this particular top event the most critical component is Pump 1 and the least critical is the Controller. The second most critical is the Pump 3 (P3) and Valve 1 (V1) is the next. Although, all

Table 32: Importance ranking for the basic events of the fuel distribution system according to the PN based method

Basic Events (BE_i)	**Pr{TE}**	**Pr{TE|$BE_{i=0}$}**	**IM (BE_i)**	**Rank**
P1	0.3150	0	0.3150	1
P2		0.2941	0.0209	4
P3		0.1736	0.1414	2
V1		0.2801	0.0349	3
V3		0.2949	0.0201	5
S1		0.3134	0.0016	6
CF		0.3147	0.0003	7

the pumps have the same failure data, they have different criticality values due to their role in the system in the context of this particular top event.

4.6.4 Quantitative Analysis using BN based Method

In order to perform the quantitative analysis of the failure behaviour of the fault tolerant fuel distribution system using the BN based method, we first need to the translate the TFT of the failure behaviour of the system into a discrete-time Bayesian Network, and then perform a query on the BN model to obtain different information regarding the dependability of the system. The Pandora TFT of Figure 92 shows the failure behaviour of Engine 1. The TFT is translated into a discrete-time Bayesian Network following the procedure shown in Section 4.4.2. The BN of the TFT is shown in Figure 95. As seen in Figure 95, the mapping from TFT to BN is one-to-one, i.e., the basic events of the TFT are mapped to the root nodes of the BN, and the intermediate events (logic gates) are mapped into internal nodes.

The prior probability table of each root node of the BN is populated based on the failure rate of the component (see Table 30) the node is representing following the procedure described in Section 4.4.2. As described in Section 4.4.2, in order to generate the prior probability values of the events, we need to define a mission time and the number of intervals *(n)* to divide the mission time into discrete intervals. There is no restriction on the maximum value of *n* but the minimum value of *n* could be equal to the maximum order of the MCSQs. The order of a minimal cut sequence (MCSQ) defines the number of basic events contribute to that MCSQ. For this particular case study, the maximum order of MCSQs is 3, so the minimum possible value of *n* is 3. The prior probability values for the root nodes of the BN of Figure 95 are calculated considering mission time as 20000 hours and dividing the mission time into 20 intervals *(n=20)*, shown in Table 33. The conditional probability table of each intermediate node of the BN is populated based on the type of TFT gate it represents following the procedure described in Section 4.4.2.

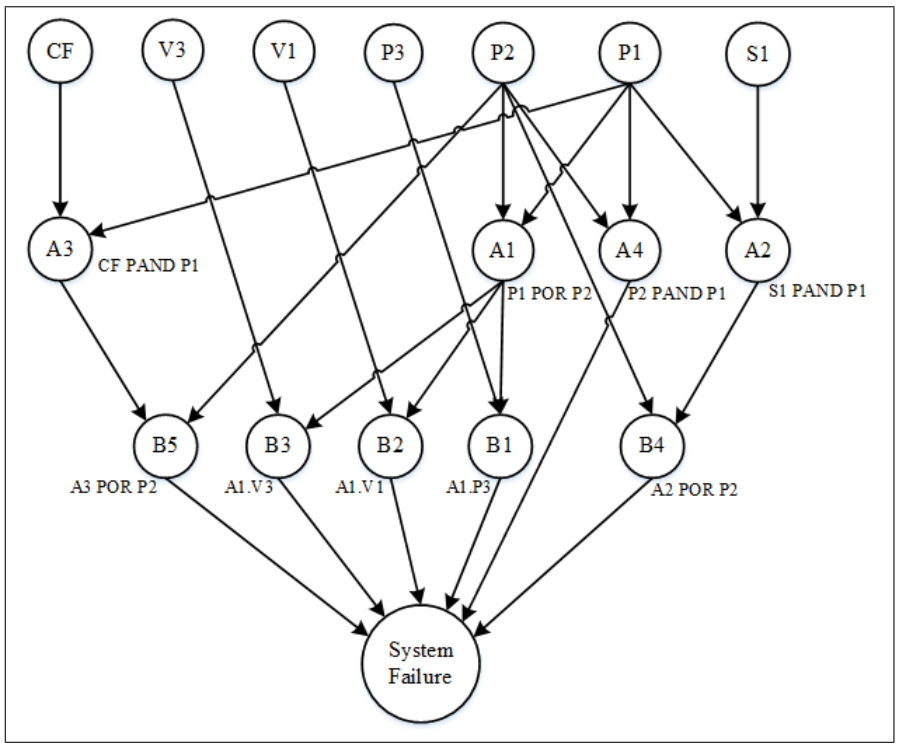


Figure 95: Bayesian Network of failure behaviour of Engine 1

Now, running a query on the node named "System Failure" would give us the value of the system unreliability for mission time 20000 hours. The value of the system unreliability obtained for 20000 hours of mission time is 0.3281 and it takes 75.8312 seconds on average to run the query. The average run time is taken as an average over 10 runs. If we choose smaller values of n then the run time will decrease. This is because of the fact that the run time depends on the number of states we need to consider for each of the nodes (events) in the BN, and the number of states depends on the value of n. As the value of n increases, number of states also increases, thus increase the execution time. However, if we consider more states for each of the nodes in the BN, then we will have a more precise value of system unreliability because of the smaller discretisation steps. Table 34 shows the top event probabilities (system unreliability) and average execution times for 20000 hours of mission time with different values of n. As seen in the table, if we increase the number of intervals (value of n) by 1, then the system unreliability is increased by a factor of approximately 1.02; on the other hand, the average execution time is increased by a factor of approximately 1.78.

Table 33: Prior probabilities of root nodes for $n = 20$ and $t = 20000$

Node	Probabilities											
	State 0	**State 1**	**State 2**	**State 3**	**State 4**	**State 5**	**State 6**	**State 7**	**State 8**	**State 9**	**State 10**	**State 11**
P1	0.5273	0.0315	0.0305	0.0295	0.0286	0.0278	0.0268	0.0260	0.0252	0.0243	0.0237	0.0228
P2	0.5273	0.0315	0.0305	0.0295	0.0286	0.0278	0.0268	0.0260	0.0252	0.0244	0.0236	0.0228
P3	0.5273	0.0315	0.0305	0.0295	0.0286	0.0278	0.0268	0.0260	0.0252	0.0244	0.0236	0.0228
V1	0.8187	0.0100	0.0098	0.0098	0.0096	0.0096	0.0094	0.0094	0.0093	0.0092	0.0091	0.0090
V3	0.8869	0.0060	0.0059	0.0059	0.0059	0.0059	0.0058	0.0057	0.0058	0.0057	0.0056	0.0057
S1	0.9512	0.0025	0.0025	0.0025	0.0025	0.0024	0.0025	0.0024	0.0025	0.0024	0.0025	0.0024
CF	0.9900	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005

Node	Probabilities								
	State 12	**State 13**	**State 14**	**State 15**	**State 16**	**State 17**	**State 18**	**State 19**	**State 20**
P1	0.0222	0.0214	0.0208	0.0201	0.0195	0.0189	0.0183	0.0177	0.0171
P2	0.0222	0.0214	0.0208	0.0201	0.0195	0.0189	0.0183	0.0177	0.0171
P3	0.0222	0.0214	0.0208	0.0201	0.0195	0.0189	0.0183	0.0177	0.0171
V1	0.0089	0.0088	0.0087	0.0087	0.0086	0.0084	0.0084	0.0083	0.0083
V3	0.0056	0.0055	0.0056	0.0055	0.0054	0.0055	0.0054	0.0053	0.0054
S1	0.0025	0.0024	0.0024	0.0024	0.0024	0.0024	0.0024	0.0024	0.0024
CF	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005

Table 34: System unreliability and average execution time after 20000 hours

n	Top Event Probability	Average Execution Time (seconds)
3	0.2742	0.0063
4	0.2900	0.0079
5	0.2995	0.0154
6	0.3058	0.0468
7	0.3104	0.1090
8	0.3138	0.2462
9	0.3164	0.4976
10	0.3186	0.9703
12	0.3217	2.9687
15	0.3249	11.9557
17	0.3264	26.6460
20	0.3281	75.8312

It is now clear that the number of intervals to divide the mission time has an effect on the system unreliability and execution time. To observe the effect of the mission time on execution time, we have performed quantitative analyses with different mission times ranging from 1000 hours to 20000 hours, and also with 6 different numbers of intervals. Table 35 shows the values of system unreliability for different mission times with different numbers of intervals. One thing to note is that to obtain the value of system unreliability for a particular mission time and a particular value of n, we need to provide a new set of prior probability values for the root nodes of the BN. For example, for 20000 hours mission time and for 6 different values of n, we provided 6 different sets of prior probability values for the root nodes of the BN, and in return we obtained 6 different values for system unreliability. The graphical view of the changes in system unreliability with mission time and different values of n is shown in Figure 96. Figure 97 shows the changes in average execution time for different mission times and different values on n. As mentioned earlier and as also seen in the graph of Figure 96, if we increase the number the intervals to divide the mission time, then the accuracy of the approximation of system unreliability increases and it appears to be converging to a ceiling value. From Figure 97, it can be seen that for a particular value of n, the average execution time remains almost the same for all values of mission time, however it increases with an increase in the value of n which is mentioned earlier. That means the mission time does not have an effect on the execution time unless we change the value of n. Therefore, the analysts have the flexibility to make a trade-off between the value of n and the execution time.

Table 35: Unreliability of the fuel distribution system for different mission with different values of n

Mission Time (Hours)	Unreliability					
	n=5	n=10	n=12	n=15	n=17	n=20
1000	0.0019	0.0019	0.0019	0.0020	0.0020	0.0020
2000	0.0071	0.0074	0.0074	0.0074	0.0074	0.0075
3000	0.0152	0.0158	0.0158	0.0159	0.0159	0.0160
4000	0.0257	0.0266	0.0267	0.0269	0.0269	0.0271
5000	0.0382	0.0396	0.0398	0.0400	0.0401	0.0403
6000	0.0521	0.0541	0.0545	0.0548	0.0550	0.0551
7000	0.0674	0.0702	0.0706	0.0711	0.0713	0.0715
8000	0.0838	0.0874	0.0879	0.0885	0.0888	0.0891
9000	0.1009	0.1053	0.1061	0.1068	0.1072	0.1076
10000	0.1187	0.1242	0.1250	0.1259	0.1264	0.1268
11000	0.1368	0.1433	0.1444	0.1455	0.1460	0.1466
12000	0.1552	0.1629	0.1642	0.1654	0.1660	0.1667
13000	0.1737	0.1826	0.1841	0.1855	0.1862	0.1870
14000	0.1922	0.2024	0.2041	0.2058	0.2066	0.2075
15000	0.2107	0.2222	0.2242	0.2261	0.2270	0.2280
16000	0.2290	0.2420	0.2441	0.2463	0.2473	0.2485
17000	0.2471	0.2615	0.2639	0.2664	0.2675	0.2688
18000	0.2649	0.2809	0.2835	0.2862	0.2875	0.2889
19000	0.2824	0.2999	0.3028	0.3057	0.3071	0.3087
20000	0.2995	0.3186	0.3217	0.3249	0.3264	0.3281

The criticality of the components of the fuel distribution system is calculated following the procedure described in Section 4.4.4 by setting the mission time to 20000 hours and n to 20. The results of the evaluation are shown in Table 36. According to the results shown in Table 36, Pump 1 (P1) contributed the most to the top event probability hence ranked as the most critical component and the lowest contribution is from the Controller (CF) which is ranked as the least critical component of the system. The second most critical is the Pump 3 (P3) and Pump 2 (P2) is the next.

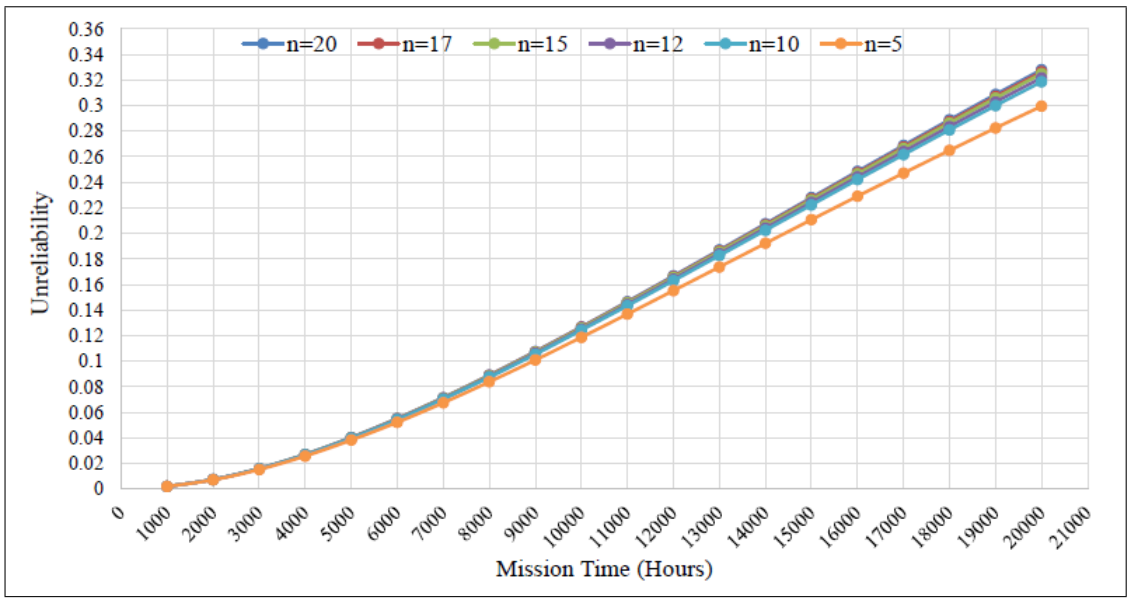


Figure 96: Changes in system unreliability with the change of values of mission time and *n*

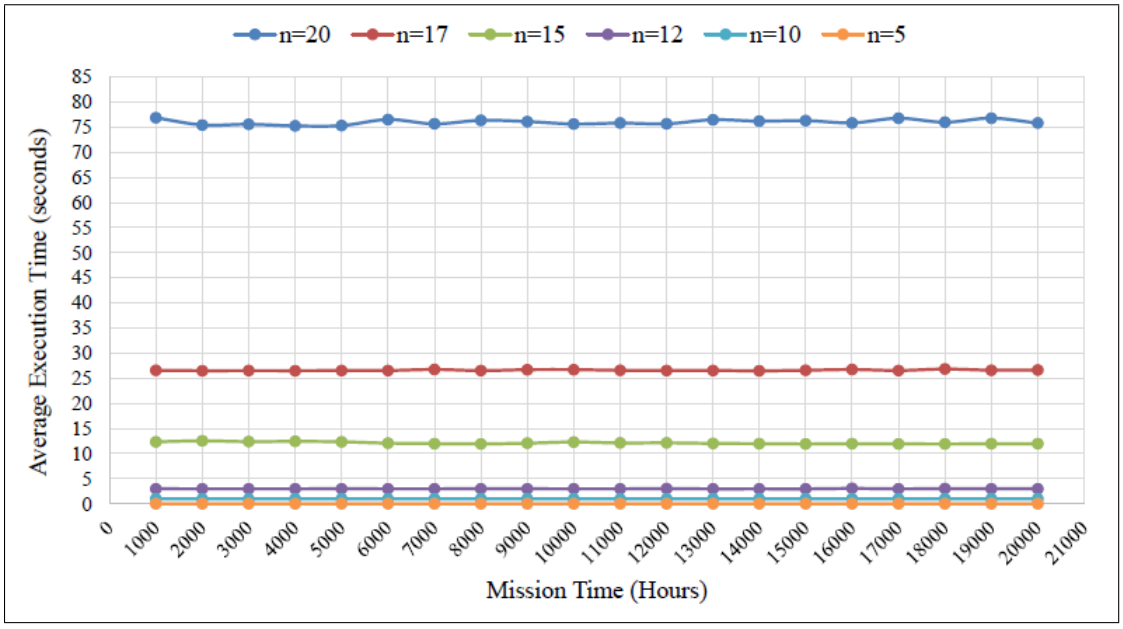


Figure 97: Changes in execution time with the change of values of mission time and *n*

We used predictive reasoning on the BN model to obtain the system unreliability by following the directions of network arcs. For predictive analysis, we use the information (failure probability) about the causes (component failure) to obtain the new belief (unreliability value) about the effect (system failure). It is effective for system analysis in the design phase because at this time analysts can provide quantitative information about system components from the historical data, if any existed at all. As the system is yet to be implemented and yet to perform its function, it is not possible to provide any definitive evidence about the occurrence of an event. However, when the system is implemented and deployed to perform its operation, then during the runtime it is possible to observe

the state of the components, and thus evidence about the occurrence of events can be obtained.

Table 36: Importance ranking for the basic events of the fuel distribution system according to the BN based method

Basic Events (BE_i)	**Pr{TE}**	**Pr{TE|BE_i = State 0}**	**IM (BE_i)**	**Rank**
P1	0.3281	0	0.3281	1
P2		0.2962	0.0319	3
P3		0.2089	0.1192	2
V1		0.2988	0.0293	4
V3		0.3112	0.0169	5
S1		0.3257	0.0024	6
CF		0.3276	0.0005	7

If we recall Section 4.4.4, then we can see that the importance measures of system component are calculated based on the evidence about the status of the system component. With the help of evidence based analysis, we can also perform diagnostic reasoning on the BN, i.e., reasoning to symptoms to causes. For example, when an analyst observes that the system has failed, based on this observation his belief about the failure probability of the components can be updated. That means a posterior probability distribution for the components is obtained given that the system has failed. Posterior probability distribution for the root nodes of the BN in Figure 95 are obtained by providing the evidence that the system has failed, and shown in Table 37. One thing to note is that this reasoning is performed backward, i.e., in the opposite direction of the arcs. Now, if we perform a predictive analysis based on this updated belief about the probability distribution of the system components, then the value of system unreliability will also be updated. For this case study, the updated value of system unreliability is 0.8315, previously it was 0.3281.

Table 37: Posterior probabilities of root nodes for $n = 20$ and $t = 20000$

Node	Probabilities											
	State 0	**State 1**	**State 2**	**State 3**	**State 4**	**State 5**	**State 6**	**State 7**	**State 8**	**State 9**	**State 10**	**State 11**
P1	0	0.0574	0.0568	0.0562	0.0556	0.0550	0.0541	0.0534	0.0526	0.0515	0.0510	0.0497
P2	0.4760	0.0423	0.0400	0.0377	0.0357	0.0338	0.0318	0.0301	0.0285	0.0269	0.0256	0.0241
P3	0.3357	0.0443	0.0429	0.0415	0.0402	0.0391	0.0377	0.0365	0.0354	0.0341	0.0333	0.0320
V1	0.7452	0.0140	0.0138	0.0138	0.0135	0.0135	0.0132	0.0132	0.0131	0.0129	0.0128	0.0126
V3	0.8411	0.0084	0.0083	0.0083	0.0083	0.0083	0.0082	0.0080	0.0081	0.0080	0.0079	0.0080
S1	0.9440	0.0034	0.0033	0.0033	0.0032	0.0030	0.0031	0.0029	0.0029	0.0028	0.0028	0.0027
CF	0.9885	0.0007	0.0007	0.0007	0.0006	0.0006	0.0006	0.0006	0.0006	0.0006	0.0006	0.0006

Node	Probabilities								
	State 12	**State 13**	**State 14**	**State 15**	**State 16**	**State 17**	**State 18**	**State 19**	**State 20**
P1	0.0491	0.0479	0.0472	0.0461	0.0452	0.0443	0.0433	0.0423	0.0413
P2	0.0229	0.0216	0.0205	0.0194	0.0184	0.0175	0.0166	0.0157	0.0149
P3	0.0312	0.0301	0.0292	0.0282	0.0274	0.0266	0.0257	0.0249	0.0240
V1	0.0125	0.0124	0.0122	0.0122	0.0121	0.0118	0.0118	0.0117	0.0117
V3	0.0079	0.0077	0.0079	0.0077	0.0076	0.0077	0.0076	0.0074	0.0076
S1	0.0027	0.0026	0.0026	0.0025	0.0025	0.0025	0.0024	0.0024	0.0024
CF	0.0006	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005

4.6.5 Quantitative Analysis using Fuzzy Set Theory Based Methodology

The fuzzy set theory based methodology is suitable to perform quantitative analysis when exact quantitative data about the system components are not known. To illustrate the idea of using fuzzy set theory based methodology described in Section 4.5 to evaluate the unreliability of the fuel distribution system, we use triangular fuzzy numbers to represent the failure possibility of basic events. The mission time is considered to be 10000 hours, so at the end of the analysis, we will obtain the fuzzy possibility and overall probability of the system failure after 10000 hours. Due to the uncertainty about the failure rate data of the basic events, experts' linguistic judgements are used to quantify the failure possibilities of the basic events. For this study, a group of six experts is constituted to provide their opinion regarding the failure possibility of the basic events and they are provided with different system related information, e.g. mission time and system architecture. The weighting scores of the chosen experts are calculated by using the data from Table 27 and shown in Table 38. Note that this scheme is used here for illustration purposes; other schemes are also possible and can be adjusted to account for opinions that carry higher weights because they are based on a stronger, more explicit and objective rationale if available. For example an expert may have consulted published reliability databases of similar components.

Table 38: Weighting scores of six experts

Expert	**Professional position**	**Experience (years)**	**Educational qualification**	**Weighting score**
E1	Professor	≥ 20	PhD	15
E2	Asst. Professor	10 *to* 14	PhD	12
E3	Engineer	5 *to* 9	M.Tech	10
E4	Manager	15 *to* 19	MSc	12
E5	Operator	< 5	Diploma	5
E6	Technician	5 *to* 9	B.Tech	8

In order to obtain the experts' opinions about the failure possibilities of the basic events as linguistic terms, seven levels of qualitative linguistic terms, i.e., Very Low (VL), Low (L), Fairly Low (FL), Medium (M), Fairly High (FH), High (H), and Very High (VH) are defined (see Table 39). The conversion scale of the linguistic terms to the fuzzy numbers are obtained using the methodology shown by Rajakarunakaran *et al.* (2015).

The opinions of the different experts regarding the failure possibility of the basic events are shown in Table 40. As the experts are different and have different backgrounds and experiences, their opinions can vary widely from one basic event to another. These

Table 39: Linguistic variables with conversion scales

Linguistic Variables	**Triangular fuzzy numbers**		
	A	**B**	**C**
Very Low (VL)	0	0.04	0.08
Low (L)	0.07	0.13	0.19
Fairly Low (FL)	0.17	0.27	0.37
Medium (M)	0.35	0.50	0.65
Fairly High (FH)	0.62	0.73	0.82
High (H)	0.81	0.87	0.93
Very High(VH)	0.92	0.96	1.0

Table 40: Expert Opinions on the Basic events

Basic Events	**Experts Opinion**					
	E1	**E2**	**E3**	**E4**	**E5**	**E6**
P1	H	FH	H	M	H	M
P2	H	FH	H	M	H	M
P3	H	H	VH	H	VH	H
V1	FH	M	FH	H	VH	M
V3	M	FH	H	FL	H	M
S1	FH	M	H	H	M	FH
CF	FL	FH	FL	FH	M	L

Table 41: Aggregation of expert opinion in triangular fuzzy form for the basic events

Basic Events	**Triangular Fuzzy Number**		
	A	**B**	**C**
P1	0.660	0.750	0.839
P2	0.660	0.750	0.839
P3	0.831	0.887	0.943
V1	0.540	0.659	0.768
V3	0.529	0.637	0.743
S1	0.603	0.708	0.806
CF	0.369	0.473	0.568

variations are accounted for by the weighting process so there is no need to discount particular values. It is therefore necessary to aggregate the results to obtain an agreement among the conflicted views of the experts. Using the methodology shown in subsection 4.5.1.3, the experts' opinions are aggregated to obtain a single consensus about the failure possibility of basic events. These aggregated results are shown in Table 41.

The aggregation process for Pump 1 is described below:
Table 42 shows the opinions of different experts about the failure possibility of Pump 1 in linguistic terms and also shows the mapping of the linguistic terms to corresponding fuzzy values. Following the procedure (Step 1: Similarity Measures) shown in subsection 4.5.1.3, the similarity matrix *SM* is formed by obtaining the similarities among the opinions. Average agreement ($AA(Ex_i)$), relative agreement degree ($RAD(Ex_i)$), weighting factor ($WF(Ex_i)$), and aggregation weight ($AW(Ex_i)$) are calculated using equations (41), (42), (43), and (44) respectively, and shown in Table 43. Now, equation (45) is used to aggregate the opinions about the failure possibility of Pump 1 and the aggregated fuzzy set for Pump 1 is $\{0.660, 0.750, 0.839\}$.

Table 42: Expert Opinion about the failure possibility of Pump 1 (P1)

Expert	**Expert Opinion**	**Fuzzy Values**		
E1	H	0.81	0.87	0.93
E2	FH	0.62	0.73	0.82
E3	H	0.81	0.87	0.93
E4	M	0.35	0.50	0.65
E5	H	0.81	0.87	0.93
E6	M	0.35	0.50	0.65

$$SM = \begin{bmatrix} 1 & 0.002088 & 1 & 0 & 1 & 0 \\ 0.002088 & 1 & 0.002088 & 0.006971 & 0.002088 & 0.006971 \\ 1 & 0.002088 & 1 & 0 & 1 & 0 \\ 0 & 0.006971 & 0 & 1 & 0 & 1 \\ 1 & 0.002088 & 1 & 0 & 1 & 0 \\ 0 & 0.006971 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Now, from the qualitative analysis, we have the minimal cut sequences (see Table 29) that are necessary and sufficient to cause the system failure, i.e., no fuel to Engine 1. From the expert opinions we have the fuzzy failure possibilities of the basic events in the triangular fuzzy form. Therefore, we can quantify the minimal cut sequences using the values from Table 41 and the fuzzy operators defined in Section 4.5.3. The fuzzy

Table 43: Different parameters for aggregation process

Expert($\mathbf{Ex_i}$)	**AA**($\mathbf{Ex_i}$)	**RAD**($\mathbf{Ex_i}$)	**WF**($\mathbf{Ex_i}$)	**AW**($\mathbf{Ex_i}$)
E1	0.400418	0.249003	0.241935	0.245469
E2	0.004041	0.002513	0.193548	0.098031
E3	0.400418	0.249003	0.161290	0.205147
E4	0.201394	0.125239	0.193548	0.159394
E5	0.400418	0.249003	0.080645	0.164824
E6	0.201394	0.125239	0.129032	0.127136

possibilities of the minimal cut sequences are shown in Table 44 and 45 respectively.

Table 44: Fuzzy possibilities of the first three MCSQs for omission of fuel to engine 1

Set	**Minimal Cut Sequences (MCSQs)**		
	$(\mathbf{P1} \wr \mathbf{P2}) \wedge \mathbf{P3}$	$(\mathbf{P1} \wr \mathbf{P2}) \wedge \mathbf{V1}$	$(\mathbf{P1} \wr \mathbf{P2}) \wedge \mathbf{V3}$
A	0.548	0.356	0.349
B	0.663	0.493	0.476
C	0.788	0.642	0.621

Table 45: Fuzzy possibilities of the second three MCSQs for omission of fuel to engine 1

Set	**Minimal Cut Sequences (MCSQs)**		
	$(\mathbf{S1} \lhd \mathbf{P1}) \wr \mathbf{P2}$	$(\mathbf{CF} \lhd \mathbf{P1}) \wr \mathbf{P2}$	$\mathbf{P2} \lhd \mathbf{P1}$
A	0.146	0.094	0.161
B	0.205	0.134	0.222
C	0.296	0.189	0.321

Using equation (53) and the fuzzy possibilities of the MCSQs from Table 44 and 45, the fuzzy possibility of the top event (no fuel to Engine 1) is obtained, which is also a triangular fuzzy number: (0.877, 0.952, 0.989).

This fuzzy possibility of the top event can be mapped to a crisp value using the equation (46), and the value calculated by equation (46) is 0.939. So, 0.939 represents the most likely possibility of the system failure after 10000 hours. This value belongs to the set Very High (VH) with 47.5% membership, which provides a good insight about the reliability of the system. If needed, we can convert this possibility value into a probability value. The value is converted to a probability value using the equation (48), and the value obtained is 0.119. So the probability of the system failure after 10000 hours is estimated to be 0.119.

The fuzzy importance measures of the fuel system components are calculated according to the method shown in Section 4.5.4. The components are ranked according to their contribution to the occurrence of the top event (system failure) and the results are shown in Table 46. As seen in the table, for the condition no fuel to Engine 1, the most critical component is Pump 1 (P1) and the least critical component is the Pump 2 (P2). The next most important are the Pump 3 (P3) and Valve 1 (V1).

Table 46: Fuzzy importance ranking for the basic events

Basic Events (E_i)	**FIM (E_i)**	**Rank**
P1	1.722	1
P2	0.018	7
P3	0.213	2
V1	0.147	3
V3	0.145	4
S1	0.103	5
CF	0.097	6

Chapter 5

Discussion and Evaluation

This chapter evaluates the contribution of this thesis and provides some recommendations for future research. Firstly, the contributions of this thesis as presented in chapters 3 and 4 are compared against the set of research questions and objectives as described in chapter 1. Secondly, the effectiveness of the technique proposed in chapter 3 in modelling and analysing dynamic behaviour of system is evaluated. Thirdly, the methodologies proposed in chapter 4 for the probabilistic evaluation of Pandora TFTs are compared with each other and also with the existing techniques. Finally, the limitations of this research are discussed and some suggestions on the direction of future research are provided.

5.1 Evaluation of Contributions Against Objectives

The first research question of this thesis states that:

How can we model the dynamic behaviour of systems and link it to dynamic analysable models that can be used by compositional MBSA techniques to perform dynamic dependability analysis?

The following three objectives were derived to address this question.

Objective 1: Identify the complications faced by the MBSA techniques due to the dynamic behaviour of systems

The intention behind this objective was to understand the dynamic behaviour of system and subsequently identify the difficulties encountered by the MBSA techniques while analysing such dynamic behaviour. This objective is achieved in section 3.2.

Firstly, the dynamic behaviour of systems is explained with reference to functionality provided by systems and different possible system configurations to provide these functionality. It was emphasised that a system component could either act as an active or a passive component. A component performs useful system operations when it acts as

an active component, on the other hand, the component remains idle and involves in no useful system operation when it acts as a passive component. A system can behave dynamically in different ways. For example, it may provide a variable set of functionality with a variable set of configurations which are obtained by selectively activating and/or deactivating some components. In this case, both system configurations and functionality change dynamically at different point in time. On the other hand, the system can provide fixed set of functionality with different configurations. In this case, the system can sacrifice some of its non-critical functionality due to the failure of some components and continues operating in a degraded mode. All these scenarios of dynamic system behaviour were explained with examples.

It was shown that this dynamic behaviour complicates the dependability analysis process for compositional MBSA techniques and makes the result inconsistent. Due to a variable number of configurations and multiple modes of operation, it is difficult for the MBSA techniques to precisely define the nominal as well as the potential failure behaviour. At the same time, as the components are activated and deactivated selectively to enter into different modes, it is a challenge for the compositional MBSA techniques to take this mode dependent behaviour into account and represent it in an understandable and manageable format.

Objective 2: Provide a methodology to represent dynamic system behaviour

As already mentioned, it is a challenge for the compositional MBSA techniques to represent mode based system behaviour and associate the mode with the failure behaviour of components. This objective therefore aimed to find a way to represent mode based dynamic behaviour in an understandable and manageable format. To achieve this objective, a state-transition based methodology, mode charts, has been identified as a potential way of representing dynamic behaviour. To define the mode based behaviour, three types of modes: *functional, degraded,* and *failed* modes are used. Each of these modes are designed to represent a distinctive system configuration. *Functional* modes represent system configurations where all the expected functionality of the system are provided, whereas *degraded* modes correspond to system configurations where some of the system functionality are sacrificed but the system is operating safely. A configuration where all or some critical set of functionality are lost is represented as a *failed* mode. Transitions from one mode to another occur due to the failure of components, hence, the transition conditions are represented as the Boolean expressions containing the failure events.

In compositional MBSA techniques, components are usually annotated with a single set of failure behaviour. However, in dynamic systems, this kind of annotation is often misleading and contributes to the inconsistency of results as mentioned earlier. In objective 1, it was identified that to perform dynamic analysis it is necessary to annotate the system component with mode based behaviour. This thesis assumes the system compo-

nent is non-repairable and provides a way to annotate the system component with mode based behaviour. Failure behaviour of a component is defined for those modes in which the component has some useful activity and the annotation data are represented in a tabular form. The tabular representation makes the data more manageable and understandable.

Using the mode chart, it easy to model the dynamic behaviour of systems. However, for a large and complex system, the mode chart could be too big and easily unmanageable because of the many possible modes and interactions among the modes. This thesis proposes the use of hierarchical mode charts instead of a flat mode chart for modelling behaviour of such a complex system. It defines some rules and provides guidelines on how to create a hierarchical mode chart from a flat mode chart. It illustrates the ideas using an example and also shows how the concept of hierarchical modelling can improve the manageability of the mode charts. Temporal logic is identified as an option that the analysts can consider to use whenever necessary to define dynamic behaviour. The scenarios when the use of temporal logic would be necessary and useful are illustrated with an example case study and the ways of using temporal logic to define dynamic behaviour are also shown.

Objective 3: Demonstrate the dynamic system modelling and dependability analysis process in HiP-HOPS

The methodology developed in the previous objective is not specific to any model based safety analysis technique. To illustrate the use of this methodology, the intention behind this objective is to demonstrate the dynamic system modelling and dependability analysis process using the HiP-HOPS tool. At present, the HiP-HOPS tool can perform static analysis and allows automatic generation of fault trees and FMEA. The analysis is performed is three phases: Annotation, Synthesis, and Analysis. At present, in the annotation phase, system components are annotated with failure data related to the static behaviour of the system. This thesis extends the annotation phase by annotating system components with mode based dynamic behaviour following the instructions provided by the methodology developed in objective 2.

Currently, the synthesis phase of HiP-HOPS produces classical fault trees by examining the system model and the failure behaviour of the components. Classical fault trees are suitable for representing the static failure behaviour of a system; however, they are not capable of representing the dynamic behaviour. As the annotation phase is extended by this thesis with mode-based annotations, to preserve the dynamic (sequence-dependent) behaviour in the synthesis process, this thesis extends the synthesis phase by generating temporal fault trees. This is achieved by examining the system model to see how the failure of components propagates through different modes in the mode chart to cause system failure.

At present, in the analysis phase, the HiP-HOPS tool takes the fault trees generated in

the synthesis phase and minimises them to obtain minimal cut sets. As the extended synthesis phase produces temporal fault trees, this thesis extends the analysis phase to analyse the temporal fault trees to minimise them to produce minimal cut sequences instead of minimal cut sets and also develops methodologies to perform quantitative analysis of temporal fault trees. The whole process of performing dynamic analysis using the HiP-HOPS tool is illustrated by applying the newly developed methodology on an Aircraft Fuel Distribution System.

The second research question of this thesis states that:

How can the Pandora temporal fault trees of a dynamic system consisting of components with various failure distributions be probabilistically evaluated using state-space based approaches?

The following objective was derived to address this question.

Objective 4: Develop methodologies to provide a state-space solution to Pandora TFTs

As already mentioned, Pandora TFTs capture sequence-dependent dynamic behaviour, hence the combinatorial solution techniques used for the evaluation of classical fault trees cannot be employed to quantify Pandora TFTs. Markov chains are widely used to provide a state-space solution to dynamic fault trees by generating all possible system states and stochastic transitions between states. Therefore, the primary intention of this objective is to develop methodologies to provide state-space solutions to Pandora TFTs. Two methodologies were developed to achieve this.

In the first methodology, Petri Net is used to probabilistically evaluate Pandora TFTs. In this methodology, firstly, the basic events of the TFTs are mapped to Generalised Stochastic Petri Nets (GSPNs). Secondly, the GSPN model for each of the TFT gates (both Boolean and Temporal gates) are developed by taking into account the logical specifications of the gates. As the MCSQs contain one or more basic events and logic gates, the GSPN model for each MCSQ is obtained by combining the GSPN model of the basic events and the logic gates contained in the MCSQ. Finally, the GSPN model for the top event is obtained by combining all the GSPN model of the MCSQs. The GSPN model of the top event can be simulated to obtain the probability of the top event. This methodology also provides a way to determine the criticality of the system components by calculating their relative contribution to the top event.

The second methodology proposed in this thesis is based on Bayesian Networks. In this methodology, Pandora TFTs are mapped to discrete time Bayesian Network (BN). The mapping is one to one, i.e., the basic events of the TFTs are mapped to the root nodes

of the BN and the logic gates are mapped to internal nodes. That means the number of nodes in the BN is equal to the total number of basic events and TFT gates. Once the mapping is done, prior probability values are assigned to root nodes based on the failure rate of the system components and the conditional probability values are assigned to the internal nodes based on the gate type the nodes are representing. Afterwards, queries are made on different nodes as required to obtain the probability of the system being in a certain state. This methodology provides a way to calculate the criticality of system components by putting observation on the root nodes.

The third research question of this thesis states that:

How can we incorporate the issue of uncertain component failure data in the quantification process of temporal fault trees and thereby allow probabilistic evaluation of dynamic systems in conditions of uncertainty?

The following objective was derived to address this question.

Objective 5: Develop a methodology to enable uncertain data to be used in the quantification of Pandora TFTs

The analysts can perform quantitative analysis using typical MBSA approaches once they have the failure data of system components available in their hands. However, as already mentioned, there are situations when the availability of the failure data cannot be guaranteed. Considering such situations, this objective was therefore to develop a methodology to facilitate the quantitative analysis of systems with uncertain failure data.

This objective has been achieved by developing a methodology based on fuzzy set theory, which is capable of quantifying Pandora TFTs with uncertain data. In this method, fuzzy set theory and expert elicitation is used to obtain opinions about the failure data of the components. Once the data about the system components are obtained as fuzzy numbers, to use these data in the quantification process, the mathematical formulae used in the analytical solution are modified to obtain the fuzzy operators for the temporal and Boolean gates. After that, the fuzzy numbers and the fuzzy operators of the gates are used to perform the quantitative analysis of the systems. This method also provides a way for measuring importance of system components in terms of their contribution to the system failure.

5.2 Evaluation of Techniques

In this section, firstly, an evaluation of the dynamic system modelling and analysis technique proposed in chapter 3 is made. Secondly, the methodologies proposed in chapter 4

for the probabilistic evaluation of Pandora TFTs are evaluated.

5.2.1 Evaluation of the technique for dynamic system modelling and dependability analysis

As already mentioned, many of the failure logic synthesis and analysis based compositional MBSA techniques are not capable of performing dynamic analysis of systems. The methodology proposed in chapter 3 for dynamic system modelling and dependability analysis provides a way of modelling the dynamic behaviour of systems and links it to dynamic analysable models. These models can be used by the compositional MBSA techniques to perform both qualitative and quantitative dynamic analysis. To illustrate the usefulness of the proposed approach in model-based dynamic analysis, the methodology has been demonstrated using HiP-HOPS as an example compositional MBSA approach.

At present, the HiP-HOPS tool can only analyse static systems and automatically generate fault trees and FMEA. Other than this, the HiP-HOPS tool can optimise system architecture using genetic algorithms and semi-automatically allocate safety requirements to the system components in the form of Safety Integrity Levels (SILs). The methodology proposed in this thesis enables HiP-HOPS to perform dependability analysis of dynamic systems. It also contributes towards the capability of HiP-HOPS to optimise the architecture of dynamic systems and also to allocate SILs to the components of dynamic systems.

One thing worth mentioning is that the HiP-HOPS tool now generates temporal fault trees instead of the classical fault trees. In static analysis, as a single set of behaviour is defined for a component, the failure of a component is represented as a basic event in the classical fault trees. However, in the dynamic analysis, as the components are annotated with mode based behaviour, i.e., a component could have multiple sets of behaviour defined, failure of a component can be represented as different basic events in the temporal fault trees. More specifically, if a system has n operational modes and if a component has different workloads in different modes then the component would contribute n basic events in the TFTs. In this case, the cause of failure of the component could be same in all the modes but the rate of failure would be different. The variable failure rates of a component in different modes are taken into account by the proposed method to distinguish the basic events originating from the same component. For this reason, the number of basic events may increase in the dynamic analysis compared to the static analysis.

Although mode charts are simple and easy to create, they can be too big and complex if we want to use them to represent the behaviour of large and complex systems. This is because such a system usually consists of several subsystems and many interactions are possible among them. Compiling all the information in a single mode chart requires calculating the permutation of all possible modes and defining all possible transitions among the modes. This will result in state space explosion. This thesis proposes the use of hierarchical mode charts to manage the complexity of flat mode charts and provides guidelines on how to use them. Although the use of hierarchical mode charts helps to

manage complexity of the flat mode charts, it will not solve the state space explosion problem completely. As a result, the state explosion problem would certainly limit the application of the proposed approach to use with small and medium-sized systems.

Another issue with mode charts is that they cannot be analysed directly. To produce comprehensive data about the system dependability, the mode charts need to be translated into other formats such as fault trees or Petri Nets. This thesis provides guidelines to translate the mode charts to Pandora temporal fault trees (TFTs), so that the dynamic behaviour of the system is preserved in the fault trees. The advantage of transforming the mode charts to Pandora TFTs is that it can be analysed both qualitatively and quantitatively. More importantly, the qualitative analysis of TFTs will also help to minimise complex temporal expressions by obtaining minimal cut sequences, thus making the subsequent quantitative analysis more manageable. As quantitative analysis of Pandora TFT is performed based on the result of qualitative analysis, the disadvantage of using Pandora TFTs is that analysts cannot perform standalone quantitative analysis of TFTs.

Mode charts could also be translated to dynamic fault trees (DFTs) but in this case analysts can only perform quantitative analysis of DFTs, thus cannot benefit from the qualitative analysis. Other than the fault trees, the mode charts could also be translated to other representations such as Markov chains (MCs), Bayesian Networks (BNs), and Petri Nets (PNs). Similar to DFTs, using these representations analysts can perform standalone quantitative analysis but qualitative analysis cannot be performed. In the conversion process, these approaches (MCs, BNs, and PNs) will inherit the state-space explosion problem of the mode charts and unlike the Pandora TFTs, as qualitative analysis is not performed on these representations to minimise the complexity, these representations will retain the complexity inherited from the mode charts and can make the analysis unmanageable.

The proposed methodology can provide similar functionality to many other compositional MBSA approaches but with more features. Similar to FPTN and CFT, this method allows analysts to perform static analysis, e.g. FTA, in addition, it also allows analysts to perform dynamic analysis which is not possible in CFTs and in the classical form of the FPTN. Unlike the FPTN and FPTC approaches, this method does not create separate failure models alongside the design models of the systems, rather it annotates the same system architecture models. As a result the models become more formal than a separate model for dependability analysis, which can allow automating all or some part of the dependability analysis process. The advantage of FPTN over the proposed approach is that at present FPTN provides systematic and modular notations for presenting failure behaviour of systems.

The proposed method can consider both static and dynamic system architectures whereas the FPTC considers system architecture as static. However, FPTC can consider more types of failure e.g., incompletion than the proposed approach and it can be used to determine and model failure behaviour of both hardware and software components of systems

but at present the application of the proposed approach is limited only to hardware components of systems. Another difference between FPTC and the proposed approach is that the proposed approach suggests the synthesis of temporal fault trees from the annotated system models, whereas FPTC does not suggest the synthesis of fault trees from the FPTC modules; rather it uses fixed point calculation. Some extension of FPTC also supports model checking facility for system architecture but at present the proposed method does not provide model checking facility. Therefore, if analysts have to analyse system with both hardware and software components or if they want to perform model checking on system architecture then they can choose FPTC, but the proposed method will be useful for the dynamic analysis of systems with hardware components.

In terms of dynamic analysis, the proposed approach can provide a similar capability to AADL's error model but only for systems with hardware components. Although both the approaches use state-event based methods to define the behaviour of components their ways of working are different. To perform dependability analysis using AADL, analysts have to define separate models for representing the nominal and the failure behaviour of systems, however, as already mentioned, in the proposed approach analysts do not have to create separate models for representing the failure behaviour of systems; rather they have to annotate the system architecture with failure behaviour of components. One thing both approaches have in common is that they increasingly suffer from state-space explosion while modelling a moderately complex system. AADL models could be translated to fault trees, GSPNs, FPTC models, and Altarica Dataflow etc. but at present, the behavioural models of system created using the proposed approach are translated only to temporal fault trees.

5.2.2 Evaluation of the Quantitative Analysis Techniques

This thesis advances quantitative dependability analysis by proposing two methodologies for probabilistic evaluation of Pandora Temporal Fault Trees. Another major contribution of this thesis is a methodology to allow the analysts to perform quantitative analysis with uncertain failure data. The effectiveness of these methodologies has already been evaluated in chapter 4 by applying them during the dependability analysis of a fault tolerant fuel distribution system of a ship. In this subsection, the proposed techniques are compared with each other and also with other existing techniques.

5.2.2.1 Petri Nets and Bayesian Networks Based Techniques

The proposed Petri Net technique transforms Pandora temporal fault trees into Generalised Stochastic Petri Nets and models time as continuous. In the transformation process, each temporal fault tree node (basic, intermediate, and top events) is mapped to a sub-net where there is a place indicating the status of the node. The failure of a component is represented as a timed transition and the failure propagation is represented as an imme-

diate transition. That means the sub-net of a TFT node could contain several places and transitions. From Figure 69, it can be seen that mapping of a basic event to a GSPN model requires three places, one timed and one immediate transitions. Mapping of an *N* input OR gate to a GSPN model would require *(N+1)* places and *N* immediate transitions (see Figure 71). On the other hand, an *N* input PAND gate could be transformed into a GSPN model by using *2N* places and *N* immediate transitions (see Figure 72). The number of arcs (directed and inhibitor) required in the sub-net of a TFT node vary widely. Due to the requirement of different number of places, transitions, and arcs to form a TFT node, a GSPN model of a simple TFT could be increasingly complex and thus would be difficult to create.

Once the GSPN model of a TFT is created, the model could be simulated to estimate the probability of the system being in a failed state after a certain amount of time. During the simulation, the analyst has to provide a mission time and a step size. The step size does not have any effect on the result obtained from the simulation, but it does have a direct effect on the simulation time. The smaller the step size, the longer it takes for the simulation to finish. It is intuitive that the simulation time would increase if the mission time is increased. Whatever step size is chosen, the analyst can observe the system unreliability after that amount of time. Therefore, analysts can choose mission time and step size most convenient to them by basing their choice on how long and how often they want to estimate the system unreliability.

Other than probabilistic evaluation of Pandora TFTs, the GSPN model of a TFT can be used to play the token game to verify the accuracy of the results of the qualitative temporal fault tree analysis. The idea is to put tokens in different places with different time stamps and then simulate the model to see whether the failed state of the system is reached for a selection of component failure in some particular order.

The PN based methodology also identifies the criticality of components by measuring the contribution of the components to the system failure. The contribution of a component is measured by taking the difference between system unreliability and the system unreliability with the component fully available. A component is made fully available by removing the token from the place that represents the status of the component.

Apart from the size and difficulties of creating GSPN model of a TFT, the Petri Net based method has another limitation: its application is currently restricted to the systems with exponentially distributed data.

The Bayesian Network (BN) based approach evaluates Pandora TFTs by translating them into discrete time BNs where each root node of the BNs represents a basic event and each internal node represent a logic gate. The transformation of a TFT gate to a BN model is simpler than that of transforming it to a GSPN model in the sense that in this case the transformation is one to one. For example, to transform an *N* input logic gate of any type to a BN we will need *N* root nodes each representing an input event and an intermediate node to represent the output of the gate. For this reason, in terms of creating

graphical models (either PN or BN) by transforming TFTs, the BN based method is less complex than the PN based method.

In general, the main challenge of creating a BN of a system model lies in defining parent child relationship between the nodes. However, the BN based methodology proposed in this thesis does not have to face this challenge because it simply has to translate TFT expressions (combination of logic gates and input events) to BN where the parent-child relationship is already defined by the qualitative analysis of the TFTs. Another issue with BN is that it is possible to create different BN models for the same system which will produce the same result, except that they differ in complexity. This is primarily a problem when parent-child relationship among the nodes are not properly defined, thus producing a complex model instead of a simple one. Again, the methodology proposed in this thesis does not suffer from this issue because of the well-defined parent-child relation in the logical expressions. As this methodology is essentially modelling some logical expressions in the form of a BN, it will always produce the same result for logically equivalent expressions but with different execution time depending on the complexity of the expressions. For example, the BN models of an un-minimised TFT and its equivalent minimised TFT should be logically equivalent, i.e., they will produce same results but the former BN model should be more complex than the latter model, just as the un-minimised TFT is more complex than the minimised TFT.

Once the graphical BN model of a TFT is created, the next step is to divide the mission time into a number of discrete intervals where the duration of intervals are same. This is similar to defining step size in the PN based method. Once the number of intervals *(n)* is decided, the prior probability values for each of the root nodes in different interval is formulated based on the failure rate of the components that the node represents. At the same time, conditional probability values for the internal nodes are defined based on the type of logic gate they represent. Unlike, the PN based method, this method can work with both exponentially and non-exponentially distributed data. A change in the number of intervals or in mission time would require the prior and conditional probability values redefined.

In the BN based method, system unreliability is obtained by performing a query on the node representing the top event of the TFT. This approach calculates the criticality of system component using the similar method used in PN based approach. The difference is that in this case, a component is made fully available by observing the BN node which represents the component to be in the State 0 (False). In addition to predictive analysis, the BN based method also allows the analyst to perform diagnostic analysis. In diagnostic analysis, if an analyst has an evidence that the system has failed then based on this evidence the failure probability of system components are updated.

As already seen in chapter 4, the number of intervals has an effect on the time required to run a query. The higher the number of intervals the more the time it requires to run a query; however, it will also increase the accuracy of the estimation. One important thing

to note is that, for a particular number of intervals, whatever mission time we choose the average execution time remains almost the same (see Figure 97). But in the PN based method, for a fixed step size, a rise in the mission time increases the execution time.

Under the same conditions, both the BN and the PN based methods take approximately equal amounts of time to estimate the system unreliability, and the values estimated are very close to each other. For example, for 20000 hours mission time, setting the step size equal 1000 hours in the PN based method is equivalent to setting *n=20* in the BN based method. Under the above conditions, the system unreliability estimated by the PN and BN based methods are 0.3150 and 0.3281 respectively; and the time taken by the two approaches are 79.40 and 75.83 seconds respectively.

In summary, the BN based method has three distinctive advantages over the PN based method:

1. A BN model for a TFT is easier to create compared to the PN model of the same TFT.

2. The BN based method is applicable to failure data with any distributions whereas the application of PN based method is limited to exponentially distributed data.

3. The BN based method allows us to perform both predictive and diagnostic analyses.

In contrast, the PN based method has two advantages over the BN based method:

1. It allows us to verify the accuracy of the qualitative results, i.e., correctness of the minimal cut sequences.

2. From a single simulation, the PN based method allows us to estimate system unreliability/reliability for multiple mission times, and thus eliminates the need for running separate simulations for different mission times. In the BN based method, to estimate system unreliability for multiple mission times, a query is required to be run for each mission time; and most importantly, for each mission time the analyst has to provide a new set of prior probability data for the root nodes.

5.2.2.2 Evaluation of PN and BN based techniques against existing techniques

The Analytical solution to Pandora TFTs uses algebraic expressions to probabilistically evaluate the temporal gates. This approach can be applied to evaluate MCSQs with any number of temporal gates by forming the algebraic expressions for each of the gates contained in the MCSQs. As a result, as the order (number of events) of a MCSQ increases, the complexity of the mathematical expressions increases accordingly. Therefore, it is difficult to form the mathematical expressions for complex MCSQs and thus their quantification using analytical approach becomes harder. Another issue with the analytical approach is that it is restricted to the exponential distribution, i.e., it can only applied to analyse systems with exponentially distributed failure data.

The simulation approach to quantify temporal fault trees uses Monte Carlo Simulation to evaluate system unreliability. This approach alleviates the limitations of the analytical approach in terms of the distribution of data and is equally applicable for exponentially and non-exponentially distributed data. In this approach, generic Monte Carlo Simulation models for each of the temporal gates has been proposed and MCSQs are evaluated by evaluating each individual gate contained in the MCSQs separately. For this reason, this approach can take too long to provide a solution to more complex TFTs.

Markov chains are widely used in the probabilistic evaluation of dynamic fault trees. In Markov chain based approaches, Continuous Time Markov Chains (CTMCs) are created by generating all possible reachable system states and stochastic transitions among states. After forming the graphical model, ordinary differential equations for all the states are derived and subsequently solved to evaluate system reliability. Similar to DFTs, Pandora TFTs can also be solved using Markov chains. Similar to the analytical solution of Pandora TFTs, the application Markov chain based approach will also be limited to system with exponentially distributed data.

As the underlying reachability graphs of stochastic Petri Nets are isomorphic to CTMCs, Petri Nets are used in this thesis to provide a state space solution to Pandora TFTs. This approach proposes Petri Nets for all the temporal and Boolean gates. The PN model of the MCSQs are formed from these, and subsequently the model is simulated to estimate system reliability. Similar to the analytical and Markov chain based quantification approaches for Pandora TFTs, this approach can only be applied to analyse system with exponentially distributed data. But the advantage of this approach over other approaches is that it can evaluate the correctness of the result of the qualitative analysis, i.e., verify that the MCSQs obtained from the qualitative analysis lead to system failure. This is done by performing a reachability analysis.

To allow the analysts to evaluate the dependability of dynamic systems featuring various failure distributions, this thesis proposes another method based on Bayesian Networks to provide a state space solution to Pandora TFTs. Unlike the analytical, Markov chain and the Petri Net based methodologies, this approach can work with both exponentially and non-exponentially distributed data. In this method, Pandora TFTs are mapped to Bayesian Networks and then the BNs are solved to perform predictive analysis of system like other approaches, i.e., estimates system unreliability using the failure data system components. Unlike other approaches, this approach also allows diagnostic analysis of systems, i.e., facilitate reasoning to symptoms to causes, such as when an analyst observes that the system has failed then based on this observation his belief about the failure probability of the components are updated.

5.2.2.3 Evaluation of Fuzzy Set theory based approach with fixed value based techniques

All the existing techniques for quantitative evaluation of Pandora temporal fault trees, including the PN and BN based methods proposed in this thesis, rely on component failure data to probabilistically evaluate system reliability. That means these approaches cannot be applied for quantitative analysis of systems if one or more system components have missing or limited failure data. The fuzzy set theory based approach alleviates the above problem by combining expert opinion and fuzzy set theory to allow quantitative, dynamic analysis when there is a scarcity of exact failure data of the components. For this reason, this approach is particularly suitable for performing system analyses in the early design phase when the exact choice about the components may have yet to be made and thus exact values for the failure data may not be known. To verify the effectiveness of the fuzzy set theory based method in estimating system unreliability with uncertain data, a comparison is made between the system unreliability values estimated by the other techniques and the value estimated by the fuzzy set theory based method, shown in Table 47. The data shown in the table are for the fault tolerant fuel distribution system used in chapter 4, all considering mission time as 10000 hours. The Difference column of the table shows with what percentage the value estimated by the proposed fuzzy approach deviates from the values estimated by the other approaches.

Table 47: Comparison of system unreliability estimated by other approaches with the unreliability estimated by the fuzzy set theory based approach

Approaches	**Unreliability**		**Unreliability estimated by the fuzzy approach**	**Difference**
Analytical	0.135		0.119	11.85% lower
PN Based	0.117			1.71% higher
BN Based	with 3 intervals	0.111		7.21% higher
	with 4 intervals	0.116		2.59% higher
	with 5 intervals	0.119		same
	with 6 intervals	0.121		1.65% lower
	with 7 intervals	0.122		2.46% lower
	with 8 intervals	0.123		3.25% lower
	with 9 intervals	0.124		4.03% lower
	with 10 intervals	0.124		4.03% lower

Although there are some small differences between the value estimated by the fuzzy set based method and the values estimated by other fixed value based methods, the im-

portant thing to note that the fuzzy temporal fault tree analysis enables the analysts to perform reliability analysis of dynamic systems in the presence of uncertain failure rate data of system components while still yielding a reasonably useful result. In particular, one important aspect of quantitative reliability analysis of system designs is to identify the critical components so that the designers can decide where to focus their efforts on those parts of the system that require most improvement to satisfy the requirements, e.g., by applying fault tolerance strategies. As seen in Table 46, the fuzzy set theory based method can also determine the criticality of components, and thus allows analysts to see the relative contribution of different system elements to the overall failure without relying on an accurate estimation of the system failure probability.

5.3 Limitations and Future Research

Although the research presented in this thesis has made some significant contributions in the model-based safety assessment domain, there remain a number of areas for further research. Below are some limitations that open new avenues for further research.

5.3.1 Reparability of Events

In this thesis, it is assumed that the system components are non-repairable. This means that once a component fails it remains in the failed state forever and is not repaired. However, in many real life systems, it is possible to repair a failed component to restore its functionality. Therefore, one possible avenue worthy of further research is the inclusion of the reparability of components in the analysis — though it will complicate the dynamic analysis to a great extent. This will definitely improve the practicability of the research presented in this thesis.

5.3.2 Dynamic System Modelling and Analysis Technique

The major limitation of the approach proposed in chapter 3 for dynamic system modelling and analysis is that it suffers from state-space explosion while modelling the behaviour of a complex system. At present the approach defines mode-based dynamic behaviour for all the system components. However, a system may contain subsystems or components which only have a single set of behaviour throughout the system lifecycle, hence do not require dynamic behaviour defined for that parts of the system. Therefore, in future, it is worth trying to explore alternative options for modelling system behaviour to minimise the state-space explosion problem. One potential option is to use separate static and dynamic modules during modelling the behaviour of the systems. Another option could be to decompose larger systems into smaller parts, analyse those parts separately, and then combine the results of those smaller analyses to obtain information about the overall dependability of the whole system.

Although the proposed approach is not aimed at any particular compositional MBSA technique, in this thesis, the utility of the approach in dynamic dependability analysis is demonstrated using the HiP-HOPS tool. Therefore, in future, it is worth performing research to apply the proposed method to other compositional MBSA approaches. At the same time, in the proposed approach, temporal fault trees are synthesised and analysed to evaluate the dependability of systems. In future, research could be performed to generate other dependability analysis artefacts such as dynamic fault trees, FMEAs, and Markov chains etc. from the behavioural system models to evaluate the dependability of the systems.

5.3.3 State Space Based Quantification Methods

To accurately capture the dynamic behaviour of systems, like other dynamic analyses approaches, Pandora has to remember the state of the system components to represent the sequencing among the occurrence of the components' failure. As the number of states increases exponentially with the number of system components, the state awareness property poses the state explosion problem for Pandora. As both the Petri Net and the Bayesian Network based methods provide state space solutions to Pandora temporal fault trees and these methods kept all the properties of Pandora unchanged, they also inherit the state explosion problem. This problem limits the applicability and the scalability of the approaches for the analysis of complex systems. As Pandora temporal fault trees consist of both Boolean and temporal gates, it would be a worthwhile effort to develop a modularisation technique that will enable the analysts to use combinatorial solutions for modules with Boolean gates and state space solutions for modules with temporal gates, thus providing a more scalable solution for dynamic system analysis.

At present, the application of the Petri Net based method is limited to systems with exponentially distributed data. Although the Bayesian Network based method alleviates this problem, it is still worth performing research to make the PN based method equally applicable for exponentially and non-exponentially distributed data.

5.3.4 Fuzzy Set Theory Based Method

The aim of the fuzzy set theory based methodology is to allow quantitative, dynamic analysis when there is a scarcity of exact failure data of the components. The first thing to note about the fuzzy set theory based method is that it is based on the analytical solution of the Pandora TFTs. As the application of the analytical solution of TFTs is limited to the system with exponentially distributed failure data, the fuzzy set theory based approach will also only be applicable to the systems with exponentially distributed failure data. Therefore, in future, it is worth trying to explore alternative options to allow the use of non-exponentially distributed failure data in the quantification process of Pandora Temporal Fault Trees with uncertain data. One potential option is to extend this work by defining

the fuzzy operators for non-exponentially distributed data by modifying the operators defined in this thesis. As already mentioned, the Bayesian Network based method is capable of performing quantitative analysis with any type of failure distribution, so another option could be to incorporate the uncertainty aspect of the failure data in the Bayesian Network based approach. However, there is of course a possibility to develop entirely different approaches to allow quantitative analysis of Pandora TFTs with uncertain data.

Chapter 6

Conclusion

Safety-critical systems are an integral part of our life but this means that the cost is significant if they fail. Our high dependence on such systems make it a priority for engineers to ensure their dependability by identifying the potential risks they pose as early as possible and then minimising the likelihood of these risks. Many classical analysis techniques such as Fault Tree Analysis (FTA) and Failure Modes Effects and Criticality Analysis (FMECA) are available to aid the system analysts in the dependability analysis processes. However, the manual nature of these classical analysis techniques makes the system analyses process time consuming and expensive. At the same time, as the analyses are performed on informal system models, these processes can result in inconsistencies and discrepancies leading to inaccurate evaluation of system dependability.

Over the past twenty years, researchers have made continuous efforts to simplify the dependability analysis process by automatically synthesising dependability related data from system models. This has led to the emergence of the field of model-based safety assessment (MBSA). In MBSA, analyses are performed on formal system models and several tools and techniques have emerged to aid the analyses. Many MBSA techniques use classical approaches like FTA as the engine for their analyses except that they generate them automatically from system model, thus alleviating many of the limitations of classical approaches.

Using compositional MBSA techniques, analysts can determine how a single component failure or combinations of failures of different components can lead to system failure. As systems grow more complex and their behaviour becomes more dynamic, assessing the effects of combinations of failure events is not enough by itself to capture the system failure behaviour; in addition, understanding the order in which they fail is also required for a more accurate failure model. Therefore there are many challenges and developments in the MBSA paradigm that limits the efficacy and application of compositional MBSA techniques to dynamic systems. To address some of these challenges, this thesis introduces a methodology to annotate system components with dynamic dependability related information using a state-event formalism based approach, thereby allowing the dynamic MBSA of systems.

Pandora, a recent dynamic extension to classical fault trees, is also integrated with model-based design and analysis. It introduces three temporal gates to allow MBSA techniques to synthesise and analyse temporal fault trees from behavioural models of the dynamic systems. Pandora's strength lies in its temporal laws, which allow qualitative analysis of dynamic systems. By performing qualitative analysis, Pandora can create useful insight to system failure in situations with limited or absent quantitative failure data, e.g. in the case of new system components. However, it has limited capability to perform quantitative analysis, which provides a measure of how reliable a system is and which parts of the system needs more attention to improve reliability. Although recent approaches have been developed to quantify Pandora temporal fault trees, these approaches have some limitations, such as being limited only to exponentially distributed data or having excessive execution times.

Given the increasing importance of model-based design and analysis, and the potential use of Pandora in this context, it is fruitful to explore possible ways to improve quantitative analysis of Pandora TFTs. Such techniques should be able to provide a state space solution to TFTs in a reasonable time, and most importantly they should be equally applicable to both exponentially and non-exponentially distributed data.

Uncertainty in failure data is yet another important area of MBSA that could affect the quantitative analysis, and surprisingly it has received little attention. In order to perform quantitative analysis, typical MBSA approaches take it as guaranteed that the failure data of system components will always be available. However, as already described in chapters 1 and 2, that there are cases where no guarantee can be given about the availability of the failure data. In such cases, the scarcity of data could force the MBSA techniques to discontinue the quantitative analysis, thus undermining the efficacy and applicability of MBSA approaches for quantitative analysis of dynamic systems. This issue motivates one of the goals of this thesis to develop a methodology to allow MBSA approaches to perform dynamic, quantitative analysis under uncertainty.

In this thesis, a number of methodologies have been developed to achieve the objectives stated in chapter 1. The contributions of this thesis can be summarised as follows:

- The complicacies faced by the compositional MBSA techniques while modelling and analysing dynamic behaviour are identified. Different state based methodologies that are capable of modelling dynamic system behaviour are studied and mode charts have been selected as a potential way of representing dynamic behaviour. Subsequently, an approach has been proposed for annotating system component with dynamic behaviour using state-event automata, thus allowing MBSA techniques to perform dynamic analysis based on these annotations.

- Considering the complexities of the mode charts that may arise while modelling large and complex systems, guidelines have been provided on potential use of hierarchical system modelling to alleviate the complexities. The necessity of temporal

logic in defining dynamic behaviour of systems has been demonstrated by an example case study, but it is not something the analysts are forced to use all time.

- The proposed dynamic modelling approach has been demonstrated using HiP-HOPS as an example MBSA tool. This has been done by extending the existing phases of the tool. The annotation phase has been extended by annotating system components with mode-based dynamic behaviour. The synthesis phase has been extended by providing ways to generate temporal fault trees by examining the system model and how the failure of components propagates through different modes in the mode chart to cause system failure. Finally, in the analysis phase, minimal cut sequences are generated by analysing the temporal fault trees. As a whole this extension of HiP-HOPS allows it to model more complex dynamic scenarios and perform compositional dynamic dependability analysis of complex systems by generating Pandora temporal fault trees (TFTs). As a result, in addition to dynamic dependability analyses, this methodology contributes towards unlocking all the features of HiP-HOPS such as architecture optimisation and safety requirements allocation for dynamic systems.

- A method based on Petri Nets (PNs) has been developed for probabilistic evaluation of Pandora TFTs. This method provides ways to transform Pandora TFT's basic events and logic gates (both temporal and Boolean) into GSPN models. These GSPN models are then combined together to form the GSPN models of MCSQs which are then combined to form the GSPN model of the top event of the TFTs. After that, the model is simulated to obtain the probability of the top event. The PN based method also allows analysts to determine the criticality of system components. In addition to the quantification of Pandora TFTs, this method can also verify the correctness of the MCSQs.

- A Bayesian Networks (BNs) based method has been developed to quantify Pandora TFTs. Similar to the PN based method this method can evaluate system reliability by quantifying Pandora TFTs and can also determine criticality of system components. However, this approach works by transforming TFTs into BNs, which confers two advantages over other approaches. Firstly, in addition to predictive analysis, it also allows diagnostic analysis. Secondly, it can be applied to system with both exponentially and non-exponentially distributed data. This therefore improves the analysis capabilities of Pandora.

- A methodology has been proposed by combining fuzzy set theory and expert elicitation to allow the analysts to perform quantitative analysis with uncertain failure data. In the case of limited availability of failure data, this method obtains opinion from experts about the failure data of components in linguistic terms and then represents them as fuzzy numbers. As a part of this method, fuzzy operators for all the logic gates of Pandora have been developed. Fuzzy failure data of components

are used by fuzzy operators of the logic gates to evaluate the system unreliability. This methodology can also determine the criticality of system components based on their contribution to the occurrence of the top event. By more explicitly highlighting areas of uncertainty in the failure data, this method can lead to a more effective quantification of uncertainty in dynamic systems, producing more realistic and robust results that can help to avoid mistaken assumptions and potential over/under estimations of system reliability

Although this thesis has made some significant contributions to the area of MBSA, there remain some limitations that open new avenues for further research. At present, the PN and the BN based method provide state space solutions to Pandora TFTs by generating all possible reachable system states and transitions between states. This could be computationally expensive since the number of paths through the state machine increases exponentially with the number of states, i.e., that the number of possible sequences of events increases dramatically. A possible improvement is to develop a modularisation technique to provide a scalable solution to temporal fault trees.

Although the fuzzy set theory based method proposed in this thesis enables us to draw helpful conclusions about the dependability of the system even in the absence of concrete failure data, its main utility lies in drawing relative (not absolute) comparisons, e.g. about the relative criticality of components. It is important to emphasise that the results can only be as reliable as the input data and the inclusion of fuzzy data cannot create accuracy where none previously existed. At present, the methodology proposed to work with uncertain data is applicable only to exponentially distributed data. Therefore, it is worthwhile trying to develop novel methodologies or improve the existing approach to work with non-exponentially distributed uncertain data.

Another area requiring improvement is the inclusion of repairable events in the dynamic analysis. At present, only the non-repairable components are considered in this thesis, yet in many real world systems components can be repaired. Therefore, it is a worthwhile try to develop methodologies to include the reparability of components when carrying out dynamic analysis.

Closing this thesis, we want to emphasise that MBSA is an emerging field which provides tools and techniques to automate the dependability analysis process, but there are many future developments and challenges remain in this area. This thesis has discussed some of these challenges and proposed approaches to alleviate them. Several new generic methodologies have been proposed to expand our ability to perform dependability analysis of dynamic systems, and they have been demonstrated in relation to the HiP-HOPS approach in particular, helping to bring its other capabilities for optimisation and allocation of safety requirements to a wider range of systems. The proposed approaches — which utilise a range of techniques, from Petri Nets to Bayesian Networks to state charts to fuzzy set theory — are used collectively to enhance the functionality of the Pandora temporal fault tree approach and would be useful to any dependability engineers who

wish to determine the reliability of state-of-the-art dynamic systems, even systems with uncertain or absent failure data. This latter capability in particular is important, as little attention has so far been paid to analysing systems under conditions of uncertainty. .

The overall result of this work is a significant step forward in increasing our understanding and ability to analyse safety-critical systems using MBSA techniques. Given our reliance on such systems, new techniques are always required to keep pace with their continual growth in complexity and increasingly dynamic behaviour. It is our hope that the work presented in this thesis can contribute to ensuring that developments in safety-critical systems do not outpace our ability to ensure their dependability.

References

Adler, R., Domis, D., Höfig, K., Kemmann, S., Kuhn, T., Schwinn, J.-P. and Trapp, M. 2011. Integration of Component Fault Trees into the UML. In: *Workshops and Symposia at MODELS*. pp. 312–327.

Åkerholm, M., Carlson, J., Fredriksson, J., Hansson, H., Håkansson, J., Möller, A., Pettersson, P. and Tivoli, M. 2007. The SAVE approach to component-based development of vehicular systems. *Journal of Systems and Software* 80(5), pp. 655–667.

Andrews, J. 1998. Tutorial: Fault Tree Analysis. In: *Proceedings of the 16th International System Safety Conference*. Available at: `http://www.fault-tree.net/papers/andrews-fta-tutor.pdf`.

Aneziris, O. and Papazoglou, I. 2004. Fast Markovian method for dynamic safety analysis of process plants. *Journal of Loss Prevention in the Process Industries* 17(1), pp. 1–8.

Arnold, A., Point, G., Griffault, A. and Rauzy, A. 2000. The AltaRica formalism for describing concurrent systems. *Fundamenta Informaticae* 40(2), pp. 109–124.

Batteux, M., Prosvirnova, T., Rauzy, A. and Kloul, L. 2013. The AltaRica 3.0 project for model-based safety assessment. In: *11th IEEE International Conference on Industrial Informatics (INDIN)*. IEEE, pp. 741–746.

Bernardi, S. and Merseguer, J. 2007. A UML Profile for Dependability Analysis of Real-Time Embedded Systems. In: *6th international workshop on Software and performance*. New York: ACM, pp. 115–124.

Bieber, P., Castel, C. and Seguin, C. 2002. Combination of fault tree analysis and model checking for safety assessment of complex system. In: *Proceedings of the 4th European Depting Conference on Dependable Computing (EDCC)*. pp. 19–31.

Bobbio, A., Franceschinis, G., Gaeta, R. and Portinale, L. 1999. Exploiting Petri Nets to Support Fault Tree Based Dependability Analysis. In: *8th International Workshops on Petri Nets and Performance Models*. Zaragoza: IEEE, pp. 146–155.

Bobbio, A., Portinale, L., Minichino, M. and Ciancamerla, E. 2001. Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. *Reliability Engineering & System Safety* 71(3), pp. 249–260.

Boiteau, M., Dutuit, Y., Rauzy, A. and Signoret, J. 2006. The AltaRica data-flow language in use: modeling of production availability of a multi-state system. *Reliability Engineering & System Safety* 91(7), pp. 747–755.

Boudali, H., Crouzen, P. and Stoelinga, M. 2007. Dynamic Fault Tree analysis using Input / Output Interactive Markov Chains. In: *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Washington DC: IEEE Computer Society, pp. 708–717.

Boudali, H., Crouzen, P. and Stoelinga, M. 2010. A Rigorous, Compositional, and Extensible Framework for Dynamic Fault Tree Analysis. *IEEE Transactions on Dependable and Secure Computing* 7(2), pp. 128–143.

Boudali, H. and Dugan, J. 2005. A new bayesian network approach to solve dynamic fault trees. In: *Proceedings of Annual Reliability and Maintainability Symposium*. IEEE, pp. 451–456.

Boudali, H. and Dugan, J. B. 2006. A Continuous-Time Bayesian Network Reliability Modeling, and Analysis Framework. *IEEE Transaction on Reliability* 55(1), pp. 86–97.

Bozzano, M. and Villafiorita, A. 2003. Improving System Reliability via Model Checking: The FSAP/NuSMV-SA Safety Analysis Platform. *Computer Safety, Reliability, and Security* 2788, pp. 49–62.

Bozzano, M. and Villafiorita, A. 2007. The FSAP/NuSMV-SA Safety Analysis Platform. *International Journal on Software Tools for Technology Transfer (STTT) - Special Section on Advances in Automated Verification of Critical Systems* 9(1), pp. 5–24.

Bruns, G. and Anderson, S. 1993. Validating Safety Models with Fault Trees. In: Górski, J., ed., *SAFECOMP '93*, Springer London, pp. 21–30.

Bukowski, J. and Goble, W. 1995. Using Markov models for safety analysis of programmable electronic systems. *ISA Transactions* 34, pp. 193–198.

Cai, K.-y. 1996. System failure engineering and fuzzy methodology: an introductory overview. *Fuzzy Sets and Systems* 83(2), pp. 113–133.

Carlson, J., Håkansson, J. and Pettersson, P. 2006. SaveCCM: An Analysable Component Model for Real-Time Systems. *Electronic Notes in Theoretical Computer Science* 160, pp. 127–140.

Chen, G.-b., Yang, Z.-c. and Sun, J.-h. 2010. Safety Analysis of Complex systems based on Bayesian Networks. In: *2nd international conference on Industrial Mechatronics and Automation*. Wuhan: IEEE, pp. 92–95.

Chiacchio, F., Cacioppo, M., D'Urso, D., Manno, G., Trapani, N. and Compagno, L. 2013. A weibull-based compositional approach for hierarchical dynamic fault trees. *Reliability Engineering & System Safety* 109, pp. 45–52.

Chiacchio, F., Compagno, L., D'Urso, D., Manno, G. and Trapani, N. 2011. Dynamic fault trees resolution: A conscious trade-off between analytical and simulative approaches. *Reliability Engineering & System Safety* 96(11), pp. 1515–1526.

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(2), pp. 45–60.

Cozman, F. 2001. *JavaBayes*, [Online]. `http://www.cs.cmu.edu/ javabayes/`.

Desrochers, A. A., Deal, T. J. and Fanti, M. P. 2005. Complex-Valued Token Petri Nets. *IEEE Trannsactions on Automation Science and Engineering* 2(4), pp. 309–318.

Deza, M. M. and Deza, E. 2009. *Encyclopedia of distances*. Springer.

Doguc, O. and Ramirez-Marquez, J. E. 2009. A generic method for estimating system reliability using Bayesian networks. *Reliability Engineering & System Safety* 94(2), pp. 542–550.

Dugan, J., Bavuso, S. and Boyd, M. 1993. Fault trees and Markov models for reliability analysis of fault-tolerant digital systems. *Reliability Engineering & System Safety* 39(3), pp. 291–307.

Dugan, J., Sullivan, K. and Coppit, D. 2000. Developing a low-cost high-quality software tool for dynamic fault-tree analysis. *IEEE Transactions on Reliability* 49(1), pp. 49–59.

Dugan, J. B., Bavuso, S. J. and Boyd, M. A. 1990. Fault Trees and Sequence Dependencies. In: *Proceedings of Annual Reliability and Maintainability Symposium*. pp. 286–293.

Dugan, J. B., Bavuso, S. J. and Boyd, M. A. 1992. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability* 41(3), pp. 363–377.

Dutuit, Y. and Rauzy, A. 1996. A linear-time algorithm to find modules of fault trees. *IEEE Transactions on Reliability* 45(3), pp. 422–425.

Edifor, E., Walker, M. and Gordon, N. 2012. Quantification of Priority-OR Gates in Temporal Fault Trees. In: Ortmeier, F. and Daniel, P., eds., *Computer Safety, Reliability, and Security SE - 9*, Springer Berlin Heidelberg, vol. 7612 of *Lecture Notes in Computer Science*, pp. 99–110.

Edifor, E., Walker, M. and Gordon, N. 2013. Quantification of Simultaneous-AND Gates in Temporal Fault Trees. In: Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T. and Kacprzyk, J., eds., *New Results in Dependability and Computer Systems SE - 13*, Springer International Publishing, vol. 224 of *Advances in Intelligent Systems and Computing*, pp. 141–151.

Edifor, E., Walker, M., Gordon, N. and Papadopoulos, Y. 2014. Using Simulation to Evaluate Dynamic Systems with Weibull or Lognormal Distributions. In: *Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX*. pp. 177–187.

Ejlali, A. and Miremadi, S. G. 2004. Fpga-based monte carlo simulation for fault tree analysis. *Microelectronics Reliability* 44(6), pp. 1017 – 1028.

Esary, J. D. and Proschan, F. 1963. Coherent Structures of Non-Identical Components. *Technometrics* 5(2), pp. 191–209.

ESSaRel. 2005. *Embedded Systems Safety and Reliability Analyser,*, [Online]. Available at: `http://essarel.de`.

Fan, Y., Zhang, J., Gong, Q. and Zhu, Y. 2011. Safety analysis for complex system based on the finite state machine theory. In: *Proceedings of the 9th International Conference on Reliability, Maintainability and Safety(ICRMS)*. Guiyang: IEEE, pp. 594–598.

Fanti, M. P., Mangini, A. M., Ukovic, W., Lesage, J.-J. and Viard, K. 2014. A petri net model of an integrated system for the health care at home management. In: *IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, pp. 582–587.

Feiler, P. and Rugina, A. 2007. Dependability Modeling with the Architecture Analysis & Design Language (AADL). Tech. Rep. July, Software Engineering Institute, Carnegie Mellon University.

Feiler, P. H., Gluch, D. P. and Hudak, J. J. 2006a. The Architecture Analysis & Design Language (AADL): An Introduction. Tech. Rep. February.

Feiler, P. H., Lewis, B. A. and Vestal, S. 2006b. The SAE Architecture Analysis & Design Language (AADL) a Standard for Engineering Performance Critical Systems. In: *Proceedings of the IEEE Conference on Computer Aided Control Systems Design*. Munich: IEEE, pp. 1206–1211.

Fenelon, P. and McDermid, J. A. 1993. An Integrated Toolset For Software Safety Analysis. *Journal of Systems and Software* 21(3), pp. 279–290.

Ferdous, R., Khan, F., Veitch, B. and Amyotte, P. R. 2009. Methodology for computer aided fuzzy fault tree analysis. *Process Safety and Environmental Protection* 87(4), pp. 217–226.

Friedenthal, S., Moore, A. and Steiner, R. 2014. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann.

Fussel, J. and Vesely, W. 1972. A new methodology for obtaining cut sets for fault trees. *Transactions of the American Nuclear Society* 15(1), pp. 262–263.

Fussell, J., Aber, E. and Rahl, R. 1976. On the Quantitative Analysis of Priority-AND Failure Logic. *IEEE Transactions on Reliability* R-25(5), pp. 324–326.

Gallina, B. and Punnekkat, S. 2011. FI4FA: A Formalism for Incompletion, Inconsistency, Interference and Impermanence Failures' Analysis. In: *Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications*. Ieee, pp. 493–500.

Ge, X., Paige, R. F. and Mcdermid, J. A. 2009. Probabilistic Failure Propagation and Transformation Analysis. In: *Proceedings of 28th International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*. pp. 215–228.

German, R. and Mitzlaff, J. 1995. Transient Analysis of Deterministic and Stochastic Petri Nets with TimeNET. In: *Proceedings of the 8th International Conference on Computer Performance Evaluation, Modelling Techniques, and Tools and MMB*. Springer-Verlag, pp. 209–223.

Gmytrasiewicz, P., Hassberger, J. A. and Lee, J. C. 1990. Fault tree based diagnostics using fuzzy logic. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(11), pp. 1115–1119.

Grunske, L. 2006. Towards an Integration of Standard Component-Based Safety Evaluation Techniques with SaveCCM. In: Hofmeister, C., Crnkovic, I. and Reussner, R., eds., *Proceedings of the Second International Conference on Quality of Software Architectures (QoSA'06)*. Berlin, Heidelberg: Springer Berlin Heidelberg, vol. 4214 of *Lecture Notes in Computer Science*, pp. 199–213.

Grunske, L., Colvin, R. and Winter, K. 2007. Probabilistic Model-Checking Support for FMEA. In: *Fourth International Conference on the Quantitative Evaluation of Systems (QEST 07)*. IEEE, pp. 119–128.

Grunske, L. and Han, J. 2008. A Comparative Study into Architecture-Based Safety Evaluation Methodologies Using AADL's Error Annex and Failure Propagation Models. In: *11th IEEE High Assurance Systems Engineering Symposium*. IEEE, pp. 283–292.

Grunske, L. and Kaiser, B. 2005. Automatic Generation of Analyzable Failure Propagation Models from Component-Level Failure Annotations. In: *Fifth International Conference on Quality Software(QSIC 2005)*. pp. 117–123.

Grunske, L., Kaiser, B. and Papadopoulos, Y. 2005. Model-Driven safety evaluation with state-event-based component failure annotations. In: Heineman, G. T., Crnkovic, I., Schmidt, H. W., Stafford, J. A., Szyperski, C. and Wallnau, K., eds., *Proceedings of the 8th international conference on Component-Based Software Engineering (CBSE'05)*. Berlin, Heidelberg: Springer Berlin Heidelberg, vol. 3489 of *Lecture Notes in Computer Science*, pp. 33–48.

Güdemann, M. and Ortmeier, F. 2010. A Framework for Qualitative and Quantitative Formal Model-Based Safety Analysis. In: *Proceedings of 12th International Symposium on High-Assurance Systems Engineering (HASE)*. pp. 132–141.

Güdemann, M. and Ortmeier, F. 2011. Towards Model-driven Safety Analysis. In: *3rd International Workshop on Dependable Control of Discrete Systems (DCDS)*. pp. 53–58.

Güdemann, M., Ortmeier, F. and Reif, W. 2007. Using Deductive Cause-Consequence Analysis (DCCA) with SCADE. In: *26th International Conference in Computer Safety, Relaibility, and Security*. pp. 465–478.

Güdemann, M., Ortmeier, F. and Reif, W. 2008. Computation of ordered minimal critical sets. In: *Proceedings of the 7th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems (FORMS/FORMAT 08)*.

Guimarees, A. C. F. and Ebecken, N. F. F. 1999. FuzzyFTA : a fuzzy fault tree system for uncertainty analysis. *Annals of Nuclear Energy* 26, pp. 523–532.

Gulati, R. and Dugan, J. B. 1997. A modular approach for analyzing static and dynamic fault trees. In: *Proceedings of Annual Reliability and Maintainability Symposium*. IEEE, pp. 57–63.

Guo, H. and Yang, X. 2008. Automatic creation of Markov models for reliability assessment of safety instrumented systems. *Reliability Engineering & System Safety* 93(6), pp. 829–837.

Harel, D. 1987. Statecharts: a visual formalism for complex systems. *Science of Computer Programming* 8(3), pp. 231 – 274.

He, L.-p., Huang, H.-z. and Zuo, M. 2007. Fault Tree Analysis Based on Fuzzy Logic. In: *Proceedings of Annual Reliability and Maintainability Symposium*. FL: IEEE, pp. 77–82.

Hei, X., Chang, L., Ma, W., Gao, J. and Xie, G. 2011. Automatic Transformation from UML Statechart to Petri Nets for Safety Analysis and Verification. In: *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering(ICQR2MSE)*. Xi'an: IEEE, pp. 948–951.

Helmer, G., Wong, J., Slagell, M., Honavar, V., Miller, L., Wang, Y., Wang, X. and Stakhanova, N. 2007. Software fault tree and coloured Petri net – based specification , design and implementation of agent-based intrusion detection systems. *International Journal of Information and Computer Security* 1(1), pp. 109–142.

Henley, E. J. and Kumamoto, H. 1981. *Reliability engineering and risk assessment*, vol. 193. Prentice-Hall Englewood Cliffs (NJ).

Hollnagel, E. 1998. *Cognitive Reliability and Error Analysis Method (CREAM)*. London: Elsevier.

Horváth, A., Paolieri, M., Ridi, L. and Vicario, E. 2012. Transient analysis of non-Markovian models using stochastic state classes. *Performance Evaluation* 69(7-8), pp. 315–335.

Hsu, H.-M. and Chen, C.-T. 1996. Aggregation of fuzzy opinions under group decision making. *Fuzzy Sets and Systems* 79(3), pp. 279 – 285.

Huang, Y., McMurran, R., Dhadyalla, G. and Jones, R. P. 2008. Probability based vehicle fault diagnosis : Bayesian network method. *Journal of Intelligent Manufacturing* 19(3), pp. 301–311.

Hura, G. S. and Atwood, J. W. 1988. The Use Of Petri Nets To Analyze Coherent Fault Trees. *IEEE Transactions on Reliability* 37(5), pp. 469–474.

IEC. 2006. Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA) (IEC 60812). Tech. rep., Geneva.

ISO. 2011. ISO 26262: Road vehicles - functional safety. ISO standard, International Organization for Standardization, Geneva, Switzerland.

Jensen, K. 1996. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use.* New York: Springer, 2nd ed.

Jensen, K., Kristensen, L. M. and Wells, L. 2007. Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer* 9(3-4), pp. 213–254.

Jiang, Y.-j., Sun, Z.-q., Xie, H.-w. and Gong, E.-l. 2010. A human error probability quantification method based on CREAM+Bayes. In: *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*. Chengdu: IEEE, pp. V1–509–V1–512.

Jianzhong, Y. and Julian, Z. 2011. Application Research of Markov in Flight Control System Safety Analysis. *Procedia Engineering* 17, pp. 515–520.

Joshi, A., Vestal, S. and Binns, P. 2007. Automatic Generation of Static Fault Trees from AADL Models. In: *DSN Workshop on Architecting Dependable Systems*.

Kabir, S., Edifor, E., Walker, M. and Gordon, N. 2014a. Quantification of Temporal Fault Trees Based on Fuzzy Set Theory. In: *Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX*. Brunów: Springer International Publishing, pp. 255–264.

Kabir, S., Walker, M. and Papadopoulos, Y. 2014b. Reliability Analysis of Dynamic Systems by Translating Temporal Fault Trees into Bayesian Networks. In: Ortmeier, F. and Rauzy, A., eds., *Model-Based Safety and Assessment*, Cham: Springer International Publishing, vol. 8822 of *Lecture Notes in Computer Science*, pp. 96–109.

Kabir, S., Walker, M. and Papadopoulos, Y. 2015. Quantitative evaluation of pandora temporal fault trees via petri nets. *IFAC-PapersOnLine* 48(21), pp. 458–463.

Kaiser, B., Gramlich, C. and Förster, M. 2007. State/event fault trees—A safety analysis model for software-controlled systems. *Reliability Engineering & System Safety* 92(11), pp. 1521–1537.

Kaiser, B., Liggesmeyer, P. and Mäckel, O. 2003. A New Component Concept for Fault Trees. In: *Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software (SCS'03)*. vol. 33, pp. 37–46.

Kim, M. C. and Seong, P. H. 2006. A computational method for probabilistic safety assessment of I&C systems and human operators in nuclear power plants. *Reliability Engineering & System Safety* 91(5), pp. 580–593.

Košecká, J. 1992. Control of Discrete Event Systems. Tech. rep., Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA.

Kwiatkowska, M., Norman, G. and Parker, D. 2011. PRISM 4.0: verification of probabilistic real-time systems. In: *Proceedings of the 23rd international conference on Computer aided verification (CAV'11)*. Springer-Verlag, pp. 585–591.

Langseth, H. and Portinale, L. 2007. Bayesian networks in reliability. *Reliability Engineering & System Safety* 92(1), pp. 92–108.

Leveson, N. G. 1995. *Safeware: System Safety and Computers*. New York, NY, USA: ACM.

Leveson, N. G. and Stolzy, J. L. 1987. Safety Analysis Using Petr Nets. *IEEE Transactions on Software Engineering* 13(3), pp. 386–397.

Li, Y. F., Huang, H. Z., Liu, Y., Xiao, N. and Li, H. 2012. A new fault tree analysis method : fuzzy dynamic fault tree analysis. *Eksploatacja i Niezawodnosc-Maintenance and Reliability* 14(3), pp. 208–214.

Li, Y. F., Mi, J., Liu, Y., Yang, Y. J. and Huang, H. Z. 2015. Dynamic fault tree analysis based on continuous-time Bayesian networks under fuzzy numbers. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability* pp. 1–12.

Liang, G.-S. and Wang, M.-J. J. 1993. Fuzzy fault-tree analysis using failure possibility. *Microelectronics Reliability* 33(4), pp. 583–597.

Lin, C.-T. and Wang, M.-J. J. 1997. Hybrid fault tree analysis using fuzzy sets. *Reliability Engineering and System Safety* 58(1997), pp. 205–213.

Lowrance, W. W. 1976. *Of Acceptable Risk: Science and the Determination of Safety*. Los Altos, California: William Kaufmann, Inc.

Mahmood, Y. A., Ahmadi, A., Verma, A. K., Srividya, A. and Kumar, U. 2013. Fuzzy fault tree analysis : a review of concept and application. *International Journal of System Assurance Engineering and Management* 4(1), pp. 19–32.

Mahmud, N. 2012. *Dynamic Model-based Safety Analysis : From State Machines to Temporal Fault Trees*. Ph.D. thesis, University of Hull.

Mahmud, N., Walker, M. and Papadopoulos, Y. 2012. Compositional synthesis of temporal fault trees from state machines. *ACM SIGMETRICS Performance Evaluation Review* 39(4), pp. 79–88.

Malhotra, M. and Trivedi, K. S. 1995. Dependability Modeling Using Petri-Nets. *IEEE Transactions on Reliability* 44(3), pp. 428–440.

Manian, R., Dugan, J. B., Coppit, D. and Sullivan, K. J. 1998. Combining Various Solution Techniques for Dynamic Fault Tree Analysis of Computer Systems. In: *Proceedings of Third IEEE International High-Assurance Systems Engineering Symposium*. Washington: IEEE, pp. 21–28.

Marquez, D., Neil, M. and Fenton, N. 2008. Solving Dynamic Fault Trees using a New Hybrid Bayesian Network Inference Algorithm. In: *16th Mediterranean Conference on Control and Automation*. IEEE, pp. 609–614.

Marsan, M. A., Balbo, G., Conte, G., Donatelli, S. and Franceschinis, G. 1996. *Modeling With Generalized Stochastic Petri Nets*. West Sussex: Wiley. Available at: `http://www.di.unito.it/ greatspn/GSPN-Wiley/`.

Marsan, M. A. and Chiola, G. 1987. On Petri nets with deterministic and exponentially distributed firing times. *Advances in Petri Nets* 266, pp. 132–145.

Merle, G., Roussel, J.-M. and Lesage, J.-J. 2011. Algebraic determination of the structure function of Dynamic Fault Trees. *Reliability Engineering & System Safety* 96(2), pp. 267–277.

Merle, G., Roussel, J.-M. and Lesage, J.-J. 2014. Quantitative Analysis of Dynamic Fault Trees Based on the Structure Function. *Quality and Reliability Engineering International* 30(1), pp. 143–156.

Merle, G., Roussel, J.-m., Lesage, J.-j. and Bobbio, A. 2010. Probabilistic Algebraic Analysis of Fault Trees With Priority Dynamic Gates and Repeated Events. *IEEE Transactions on Reliability* 59(1), pp. 250–261.

Mi, J., Li, Y., Huang, H.-Z., Liu, Y. and Zhang, X. 2012. Reliability analysis of multi-state systems with common cause failure based on Bayesian Networks. In: *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE)*. Chengdu: IEEE, 2, pp. 1117–1121.

Mokos, K., Katsaros, P., Bassiliades, N., Vassiliadis, V. and Perrotin, M. 2008. Towards compositional safety analysis via semantic representation of component failure behaviour. In: *Proceedings of the Eighth Joint Conference on Knowledge-Based Software Engineering*. Amsterdam, The Netherlands: IOS Press, pp. 405–414.

Molloy, M. K. 1982. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers* c-31(9), pp. 913–917.

Montani, S., Portinale, L., Bobbio, A. and Codetta-Raiteri, D. 2008. Radyban: A tool for reliability analysis of dynamic fault trees through conversion into dynamic Bayesian networks. *Reliability Engineering & System Safety* 93(7), pp. 922–932.

Montani, S., Portinale, L., Bobbio, A., Varesio, M. and Codetta-Raiteri, D. 2006. A tool for automatically translating Dynamic Fault Trees into Dynamic Bayesian Networks. In: *Annual Reliability and Maintainability Symposium (RAMS'06)*. IEEE, pp. 434–441.

Murata, T. 1989. Petri Nets : Properties , Analysis and Applications. *Proceedings of IEEE* 77(4), pp. 541–580.

Neil, M. and Marquez, D. 2012. Availability modelling of repairable systems using Bayesian networks. *Engineering Applications of Artificial Intelligence* 25(4), pp. 698–704.

Neil, M., Tailor, M., Marquez, D., Fenton, N. and Hearty, P. 2008. Modelling dependable systems using hybrid Bayesian networks. *Reliability Engineering & System Safety* 93(7), pp. 933–939.

Niu, R., Tang, T., Lisagor, O. and McDermid, J. A. 2011. Automatic Safety Analysis of Networked Control System based on Failure Propagation Model. In: *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. pp. 53–58.

O'Connor, A. N. 2011. *Probability distributions used in reliability engineering*. RIAC.

Onisawa, T. 1988. An approach to human reliability in man-machine systems using error possibility. *Fuzzy Sets and Systems* 27(2), pp. 87–103.

Ortmeier, F., Reif, W. and Schellhorn, G. 2005. Deductive Cause-Consequence Analysis. In: *Proceedings of the 6th IFAC World Congress*. pp. 1434–1439.

Paige, R., Rose, L., Ge, X., Kolovos, D. and Brooke, P. 2008. FPTC: Automated safety analysis for domainspecific languages. In: *Proceedings of Workshop on NonFunctional System Properties in Domain Specific Modeling Languages, co-located with 11th International Conference of Model Driven Engineering Languages and Systems (MoDELS 2008)*. pp. 229–242.

Pais, R., Gomes, L. and Barros, J. P. 2011. From UML state machines to Petri nets: History attribute translation strategies. In: *37th Annual Conference of the IEEE Industrial Electronics Society(IECON)*. Melbourne: IEEE, pp. 3776–3781.

Palshikar, G. K. 2002. Temporal fault trees. *Information and Software Technology* 44(3), pp. 137–150.

Pande, P. K., Spector, M. E. and Chatterjee, P. 1975. Computerized Fault Tree Analysis: TREEL and MICSUP. Tech. rep., University Of California Operations Research Centre, California, USA.

Papadopoulos, Y. 2000. *Safety-Directed System Monitoring Using Safety Cases*. Ph.D. thesis, University of York.

Papadopoulos, Y. 2012. *HiP-HOPS*, [Online]. Available at: `http://hip-hops.eu/`.

Papadopoulos, Y. and Grante, C. 2005. Evolving car designs using model-based automated safety analysis and optimisation techniques. *Journal of Systems and Software* 76(1), pp. 77–89.

Papadopoulos, Y. and Maruhn, M. 2001. Model-Based Synthesis of Fault Trees from Matlab - Simulink models. In: *International Conference on Dependable Systems and Networks (DSN)*. pp. 77–82.

Papadopoulos, Y., Mcdermid, J., Sasse, R. and Heiner, G. 2001. Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. *Journal of Reliability Engineering and System Safety* 71(3), pp. 229–247.

Papadopoulos, Y. and Mcdermid, J. A. 1999. Hierarchically Performed Hazard Origin and Propagation Studies. In: *Proceedings of the 18th International Conference on Computer Safety, Reliability and Security*. pp. 139–152.

Papadopoulos, Y., Parker, D. and Grante, C. 2004a. Automating the Failure Modes and Effects Analysis of Safety Critical Systems. In: *Proceedings of the Eighth IEEE International Symposium on High Assurance Systems Engineering(HASE'04)*. pp. 310–311.

Papadopoulos, Y., Parker, D. and Grante, C. 2004b. A method and tool support for model-based semi-automated failure modes and effects analysis of engineering designs. In: *Proceedings of the 9th Australian workshop on Safety critical systems and software-Volume 47*. Australian Computer Society, Inc., pp. 89–95.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: Networks of Plausible Inference*. Morgan Kaufmann.

Peterson, J. L. 1977. Petri Nets. *ACM Computing Surveys* 9(3), pp. 223–252.

Platis, A., Limnios, N. and Du, M. 1998. Dependability analysis of systems modeled by non-homogeneous Markov chains. *Reliability Engineering and System Safety* 61, pp. 235–249.

Point, G. and Rauzy, A. 1999. AltaRica: Constraint automata as a description language. *European Journal on Automation* 33(8-9), pp. 1033–1052.

Price, C. and Taylor, N. 2002. Automated multiple failure FMEA. *Reliability Engineering & System Safety* 76(1), pp. 1–10.

Pukite, J. and Pukite, P. 1998. *Modeling for Reliability Analysis: Markov Modeling for Reliability, Maintainability, Safety, and Supportability Analyses of Complex Systems*. Wiley.

Rajakarunakaran, S., Kumar, A. M. and Prabhu, V. A. 2015. Applications of fuzzy faulty tree analysis and expert elicitation for evaluation of risks in LPG refuelling station. *Journal of Loss Prevention in the Process Industries* 33, pp. 109–123.

Rao, K., Gopika, V., Rao, V., Kushwaha, H., Verma, A. and Srividya, A. 2009. Dynamic fault tree analysis using Monte Carlo simulation in probabilistic safety assessment. *Reliability Engineering & System Safety* 94(4), pp. 872–883.

Rauzy, A. 2002. Mode automata and their compilation into fault trees. *Reliability Engineering & System Safety* 78(1), pp. 1–12.

Reza, H., Pimple, M., Krishna, V. and Hilde, J. 2009. A Safety Analysis Method Using Fault Tree Analysis and Petri Nets. In: *2009 Sixth International Conference on Information Technology: New Generations*. Las Vegas: IEEE, pp. 1089–1094.

RoSPA. 2011. *RoSPA Guide to Road Safety Projects*, [Online]. Available at: http://www.rospa.com/roadsafety/adviceandinformation/general/road-safety-projects.aspx.

Ross, T. J. 2004a. Development of membership functions. In: *Fuzzy Logic with Engineering Applications*, John Wiley & Sons, Ltd, pp. 174–210.

Ross, T. J. 2004b. Properties of membership functions, fuzzification, and defuzzification. In: *Fuzzy Logic with Engineering Applications*, John Wiley & Sons, Ltd, pp. 89–116.

Rugina, A., Kanoun, K. and Kaâniche, M. 2007. A System Dependability Modeling Framework Using AADL and GSPNs. In: *Architecting Dependable Systems IV*. pp. 14–38.

SAE. 2012. Architecture Analysis & Design Language (AADL) (AS5506B). Tech. rep., Society of Automotive Engineers Aerospace, Warrendale, PA.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K. and Teneketzis, D. C. 1996. Failure diagnosis using discrete-event models. *IEEE Transactions on Control Systems Technology* 4(2), pp. 105–124.

Semanderes, S. N. 1971. ELRAFT: A Computer Program for the Efficient Logic Reduction Analysis of Fault Trees. *IEEE Transactions on Nuclear Science* 18(1), pp. 481–487.

Sharvia, S., Kabir, S., Walker, M. and Papadopoulos, Y. 2015. Model-based Dependability Analysis: State-of-the-art, Challenges, and Future Outlook . In: *Software Quality Assurance: In Large Scale and Complex Software-Intensive Systems*, Morgan Kaufmann, pp. 251–278.

Sharvia, S. and Papadopoulos, Y. 2015. Integrating model checking with hip-hops in model-based safety analysis. *Reliability Engineering & System Safety* 135, pp. 64 – 80.

Singer, D. 1990. A fuzzy set approach to fault tree and reliability analysis. *Fuzzy Sets and Systems* 34(2), pp. 145–155.

Sinnamon, R. M. and Andrews, J. 1997. New approaches to evaluating fault trees. *Reliability Engineering & System Safety* 58(2), pp. 89–96.

Steiner, M., Keller, P. and Liggesmeyer, P. 2012. Modeling the effects of software on safety and reliability in complex embedded systems. In: Ortmeier, F. and Daniel, P., eds., *Computer Safety, Reliability, and Security*, Berlin, Heidelberg: Springer Berlin Heidelberg, vol. 7613 of *Lecture Notes in Computer Science*, pp. 454–465.

Suresh, P., Babar, A. and Raj, V. 1996. Uncertainty in fault tree analysis : A fuzzy approach. *Fuzzy Sets and Systems* 83(2), pp. 135–141.

Tanaka, H., Fan, L. T., Lai, F. S. and Toguchi, K. 1983. Fault-Tree Analysis by Fuzzy Probability. *IEEE Transactions on Reliability* R-32(5), pp. 453–457.

Tao, C.-C. 2009. A two-stage safety analysis model for railway level crossing surveillance systems. In: *IEEE International Conference on Control and Automation*. Christchurch: IEEE, pp. 1497–1502.

Torres-Toledan, J. G. and Sucar, L. E. 1998. Bayesian networks for reliability analysis of complex systems. In: *Progress in Artificial Intelligence—IBERAMIA 98*. pp. 195–206.

Tyagi, S. K., Pandey, D. and Kumar, V. 2011. Fuzzy Fault Tree Analysis for Fault Diagnosis of Cannula Fault in Power Transformer. *Applied Mathematics* 2(11), pp. 1346–1355.

Tyagi, S. K., Pandey, D. and Tyagi, R. 2010. Fuzzy set theoretic approach to fault tree analysis. *International Journal of Engineering, Science and Technology* 2(5), pp. 276–283.

US, D. o. D. 1980. Procedures for Performing a Failure Mode, Effects, and Criticality Analysis (MIL-STD-1629A). Tech. rep., Washington DC, USA.

Verma, A., Srividya, A., Prabhudeva, S. and Vinod, G. 2006. Reliability analysis of Dynamic fault tree models using fuzzy sets. *Communications in Dependability and Quality Management* 9(4), pp. 68–78.

Vesely, W., Dugan, J., Fragola, J., Minarick, J. and Railsback, J. 2002. Fault Tree Handbook with Aerospace Applications. Tech. rep., NASA office of safety and mission assurance, Washington, DC.

Vesely, W. E., Goldberg, F. F., Roberts, N. H. and Haasl, D. F. 1981. Fault Tree Handbook. Tech. rep., US Nuclear Regulatory Commission, Washington DC, USA.

Villemeur, A. 1991. *Reliability, Availability, Maintainability and Safety Assessment: Methods and Techniques*. Chichester: John Wiley & Sons.

Walker, M., Bottaci, L. and Papadopoulos, Y. 2007. Compositional Temporal Fault Tree Analysis. In: *Proceedings of the 26th International Conference on Computer Safety, Reliability and Security (SAFECOMP'07)*. pp. 106–119.

Walker, M., Manmud, N., Papadopoulos, Y., Tagliabò, F., Torchiaro, S., Schierano, W. and Lönn, H. 2008. ATESST2: Review of relevant Safety Analysis Techniques. Tech. rep.

Walker, M. and Papadopoulos, Y. 2009. Qualitative temporal analysis: Towards a full implementation of the Fault Tree Handbook. *Control Engineering Practice* 17(10), pp. 1115–1125.

Walker, M., Papadopoulos, Y., Parker, D., Lönn, H., Törngren, M., Chen, D., Johansson, R. and Sandberg, A. 2009. Semi-Automatic FMEA supporting complex systems with combinations and sequences of failures. In: *SAE World Congress*.

Walker, M. D. 2009. *Pandora: A Logic for the Qualitative Analysis of Temporal Fault Trees*. Ph.D. thesis, University of Hull.

Wallace, M. 2005. Modular Architectural Representation and Analysis of Fault Propagation and Transformation. *Electronic Notes in Theoretical Computer Science* 141(3), pp. 53–71.

Wan, Y. and Wu, C. 2009. Software Reliability Model Based on Stochastic Theory. In: *2nd International Conference on Biomedical Engineering and Informatics*. Tianjin: IEEE, 20070533, pp. 1–5.

Wang, D., Zhang, P. and Chen, L. 2013. Fuzzy fault tree analysis for fire and explosion of crude oil tanks. *Journal of Loss Prevention in the Process Industries* 26(6), pp. 1390 – 1398.

Wang, L. X. 1997. *A course in fuzzy system and control*. Prentice-Hall PTR.

Weber, P., Medina-Oliva, G., Simon, C. and Iung, B. 2012. Overview on Bayesian networks applications for dependability, risk analysis and maintenance areas. *Engineering Applications of Artificial Intelligence* 25(4), pp. 671–682.

Yang, L. 2011. Analysis on Dynamic Fault Tree Based on Fuzzy Set. *Applied Mechanics and Materials* 110-116, pp. 2416–2420.

Yang, Y., Zeckzer, D., Liggesmeyer, P. and Hagen, H. 2011. ViSSaAn : Visual Support for Safety Analysis. *Dagstuhl Follow-Ups* 2, pp. 378–395.

Yuge, T. and Yanagi, S. 2008. Quantitative analysis of a fault tree with priority AND gates. *Reliability Engineering & System Safety* 93(11), pp. 1577–1583.

Yuhua, D. and Datao, Y. 2005. Estimation of failure probability of oil and gas transmission pipelines by fuzzy fault tree analysis. *Journal of Loss Prevention in the Process Industries* 18(2), pp. 83–88.

Zadeh, L. 1965. Fuzzy Sets. *Information and Control* 8(3), pp. 338–353.

Zhang, P. and Chan, K. W. 2012. Reliability evaluation of phasor measurement unit using monte carlo dynamic fault tree method. *IEEE Transactions on Smart Grid* 3(3), pp. 1235–1243.

Zhang, X., Miao, Q., Fan, X. and Wang, D. 2009. Dynamic fault tree analysis based on Petri nets. In: *8th International Conference on Reliability, Maintainability and Safety(ICRMS)*. Chengdu: IEEE, pp. 138–142.

Zhong, X. and Li, Q. 2013. Component importance and sensitivity analysis in Bayesian networks. In: *2013 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE)*. Chengdu: IEEE, pp. 320–325.

Zurawski, R. and Zhou, M. 1994. Petri nets and industrial applications: A tutorial. *IEEE Transactions on Industrial Electronics* 41(6), pp. 567–583.

Zwick, R., Carlstein, E. and Budescu, D. V. 1987. Measures of similarity among fuzzy concepts: A comparative analysis. *International Journal of Approximate Reasoning* 1(2), pp. 221 – 242.