Loughborough University

This item was submitted to Loughborough's Research Repository by the author.
Items in Figshare are protected by copyright, with all rights reserved, unless otherwise indicated.

Methods for the efficient measurement of phased mission system reliability and component importance

PLEASE CITE THE PUBLISHED VERSION

PUBLISHER

© Sean Reed

LICENCE

CC BY-NC-ND 4.0

REPOSITORY RECORD

Reed, Sean. 2011. “Methods for the Efficient Measurement of Phased Mission System Reliability and Component Importance”. figshare. https://hdl.handle.net/2134/9086.

Loughborough University

This item was submitted to Loughborough's Institutional Repository (https://dspace.lboro.ac.uk/) by the author and is made available under the following Creative Commons Licence conditions.

creative commons

COMMONS DEED

Attribution-NonCommercial-NoDerivs 2.5

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:



Attribution. You must attribute the work in the manner specified by the author or licensor.



Noncommercial. You may not use this work for commercial purposes.



No Derivative Works. You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/

Methods for the efficient measurement of phased mission system reliability and component importance

By

Sean Reed

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University

December 2010

© by Sean Reed, 2010

Abstract

An increasing number of systems operate over a number of consecutive time periods, in which their reliability structure and the consequences of failure differ, in order to perform some overall operation. Each distinct time period is known as a phase and the overall operation is known as a phased mission. Generally, a phased mission fails immediately if the system fails at any point and is considered a success only if all phases are completed without failure. The work presented in this thesis provides efficient methods for the prediction and optimisation of phased mission reliability.

A number of techniques and methods for the analysis of phased mission reliability have been previously developed. Due to the component and system failure time dependencies introduced by the phases, the computational expense of these methods is high and this limits the size of the systems that can be analysed in reasonable time frames on modern computers. Two importance measures, which provide an index of the influence of each component on the system reliability, have also been previously developed. This is useful for the optimisation of the reliability of a phased mission, however a much larger number have been developed for non-phased missions and the different perspectives and functions they provide are advantageous.

This thesis introduces new methods as well as improvements and extensions to existing methods for the analysis of both non-repairable and repairable systems with an emphasis on improved efficiency in the derivation of phase and mission reliability. New importance measures for phased missions are also presented, including interpretations of those currently available for non-phased missions. These provide a number of interpretations of component importance, allowing those most suitable in a given context to be employed and thus aiding in the optimisation of mission reliability. In addition, an extensive computer code has been produced that implements and tests the majority of the newly developed techniques and methods.

Acknowledgements

Firstly, I wish to express my sincere thanks and appreciation to my supervisors, Professor John Andrews and Doctor Sarah Dunnett, for their continued guidance, encouragement and friendship throughout the programme of this work.

I would like to thank my colleagues at the Department of Aeronautical and Automotive Engineering, Loughborough University, and at the NTEC Centre, The University of Nottingham, for contributing to an enjoyable environment in which to work. I also want to thank BAE Systems for sponsoring this research.

Finally, I would like to thank my friends and family for all the support and encouragement they have given me.

Contents

List of Figures viii
List of Tables x
Nomenclature xiii
Acronyms xxii
1 Introduction 1
1.1 Common Properties of Real World Phased Mission Systems 3
1.2 Reliability Analysis of Phased Missions 5
1.2.1 Reliability Measurement 5
1.2.2 Reliability Improvement 6
1.3 Research Objectives 7
2 Reliability Theory and Definitions 8
2.1 Introduction 8
2.2 Definition of Risk 8
2.3 Components and System Definitions 8
2.4 Reliability Metrics 9
2.4.1 Item State 9
2.4.2 Time to first failure 9
2.4.3 Reliability and Unreliability 9
2.4.4 Availability and Unavailability 10
2.5 System Reliability Structure Representation 10
2.5.1 Event Trees 11
2.5.2 Reliability Block Diagrams 12
2.5.3 Fault Trees 12
2.5.4 Binary Decision Diagrams (BDDs) 13
2.6 Representing the top event in terms of basic event combinations 17
2.6.1 Cut Sets 17
2.6.2 Implicants 18
2.7 Quantification methods 18
2.7.1 Fault Tree Based Quantification Methods 18

2.7.2 BDD Based Quantification Methods 19
2.7.3 Markov Model Methods 20
2.7.4 Petri Net Models 21
2.7.5 Monte Carlo Simulation 23
2.8 Phased Mission Analysis 23
2.8.1 Phased Mission Fault Trees 24
2.8.2 Calculation of system failure during any period of the mission 25
2.9 Summary 26
3 Literature Review of reliability methods for non-repairable phased missions 27
3.1 Introduction 27
3.2 Early Phased Mission Methods 27
3.2.1 Conclusions 30
3.3 Boolean phase algebra based methods 30
3.3.1 Inclusion-Exclusion expansion based methods 31
3.3.2 BDD based methods 43
3.3.3 Summary and Conclusions 53
4 New methods for the analysis of non-repairable phased missions 55
4.1 Introduction 55
4.2 New Binary Decision Diagram based method 55
4.2.1 BDD Construction 56
4.2.2 BDD Probability Evaluation 62
4.2.3 Full Example 69
4.3 Software Implementation 71
4.3.1 Mission Reliability Analysis Overview 72
4.3.2 File Formats 73
4.3.3 Software Implementation Details 76
4.4 Comparison to method by Tang and Dugan 89
4.5 Summary and Conclusions 93
5 Literature review of reliability methods for repairable phased missions 94
5.1 Markov model based methods 94
5.1.1 Methods with a separate transition matrix for each phase 94
5.1.2 Single Transition Matrix 98
5.2 BDD based method 99

5.3 Markov Computation 104
5.4 Techniques for reducing computational expense 104
5.5 Monte Carlo Simulation based methods 105
5.6 Summary and Conclusions 105
6 New Methods for the analysis of repairable phased missions 107
6.1 Introduction 107
6.2 Method for analysing missions with multiple failure mode components 107
6.2.1 Extending the Markov method to systems with multiple failure mode components 108
6.2.2 Using BDDs to identify system failure states 109
6.2.3 Summary 111
6.3 Improved Evaluation Efficiency in Wang and Trivedi BDD method 111
6.3.1 BDD Evaluation through Repairable Implicant Tree 112
6.3.2 Summary 116
6.4 A method for analysing repairable phased mission systems that contain some non-repairable components 116
6.4.1 End of phase repaired component integration method 116
6.4.2 Method for repairable components with multiple failure modes 119
6.5 Summary and Conclusions 130
7 Literature Review of Reliability Importance Measures 132
7.1 Introduction 132
7.2 Non-Phased Missions 133
7.2.1 Example System 133
7.2.2 Single Event Importance Measures 134
7.2.3 Relationships between the importance measures 145
7.2.4 First Order Group Importance Measures 145
7.2.5 Higher Order Group Importance Measures 147
7.2.6 Conditional Importance Measures 154
7.3 Phased Missions 155
7.3.1 Example Missions 155
7.3.2 Component and system time periods in phased mission importance measures 159
7.3.3 Phased Birnbaum Importance Measure 160

7.3.4 Phased Criticality Importance Measure 166
7.4 Summary and Conclusions 175
8 Phased Mission Importance Measures 177
8.1 Introduction 177
8.2 Component and system time periods in phased mission importance measures 179
8.3 Importance Measure Definitions 180
8.3.1 Introduction 180
8.3.2 Phased Birnbaum II Importance Measure 181
8.3.3 Phased Criticality II Importance Measure 188
8.3.4 Phased DIM 191
8.3.5 Phased Risk Achievement Worth (RAW) 192
8.3.6 Phased Risk Reduction Worth (RRW) 193
8.3.7 Definitions 194
8.3.8 Phased Fussell-Vesely Importance Measure 195
8.3.9 Phased Criticality III Importance Measure 197
8.4 First Order Group Importance Measures 201
8.4.1 Phased Group DIM 201
8.5 Higher Order Group Importance Measures 203
8.5.1 Phased Group RAW 203
8.5.2 Phased Group RRW 204
8.5.3 Phased Joint Birnbaum II Importance Measure 205
8.5.4 Phased Group Criticality III Importance Measure 208
8.5.5 Phased DIM II 209
8.6 Conditional Importance Measures 210
8.7 Reliability Cost Minimisation 211
8.7.1 Reliability Costs 211
8.7.2 Overall Reliability Costs 213
8.7.3 Finding the optimum component reliability improvement 213
8.8 Summary and Conclusions 217
9 Conclusions and Areas for Further Work 219
9.1 Summary and Conclusions 219
9.2 Areas for Further Work 223

9.2.1 Analysis Methods 223
9.2.2 Computer Code 223

List of Figures

Figure 1.1 – Strategy for designing a high reliability phased mission system. 1
Figure 1.2 - Factors influencing the complexity of analysing phased mission system reliability. 2
Figure 1.3 - Factors affecting ability to analyse the reliability of a phased mission system. 2
Figure 1.4 – Photo of a retractable landing gear system from a Boeing 777-300 aircraft. 4
Figure 1.5 – Photo of a JetBlue Airways aircraft making an emergency landing after the front landing gear malfunctioned during take-off. 4
Figure 2.1 – An example of an event tree. 11
Figure 2.2 - A Simple Reliability Block Diagram 12
Figure 2.3 - A simple fault tree 12
Figure 2.4 – An example of a BDD with variable ordering: A < B < C < D. 15
Figure 2.5 - A fault tree and an equivalent BDD. 17
Figure 2.6 - Example Markov Diagram. 21
Figure 2.7 - Petri net transition 22
Figure 2.8 - Petri nets of common fault tree gates. 22
Figure 2.9 – Landing gear system altitude and reliability in a phased mission 24
Figure 2.10 – Common mission fault trees 25
Figure 3.1 - Example of a phase fault tree transformation with the EZ method 28
Figure 3.2 - Phase fault trees 38
Figure 3.3 – Reduction in BDD size through modified BDD algorithm from Zang et al. 44
Figure 3.4 – BDD construction example 1. 49
Figure 3.5 – BDD construction example 2. 50
Figure 3.6 – Labels for a BDD node's relative children 51
Figure 4.1 - Fault tree for a phased mission system. 57
Figure 4.2 - BDD calculated for phased mission shown in Figure 4.1 through Tang and Dugan's method. 58
Figure 4.3 - Example application of Equation 4.1. 60
Figure 4.4 – BDD computation result reduces from a) to b) due to reduction rule 61
Figure 4.5 – Step *A* of forming an implicant tree for an *ite* node. 65
Figure 4.6 – Step *B* of the data structure derivation. 65
Figure 4.7 – Equivalent Implicant Trees due to merging of summing nodes. 66
Figure 4.8 – BDD with nodes *A*-*F*. 67
Figure 4.9 – Implicant Trees *A*-*F* corresponding to nodes *A*-*F* from Figure 4.8. 68
Figure 4.10 - Mission failure fault tree for example phased mission. 69
Figure 4.11 – BDD representation of the phased mission for the example phased mission from Figure 4.10. 70
Figure 4.12 – Implicant Tree from from the BDD shown in Figure 4.11. 71
Figure 4.13 – Overview of steps performed to analyse reliability of a phased missions 72
Figure 5.1 – BDD node *G* and its children, G1 and G0, labelled with their state probability vectors. 102
Figure 6.1 – An example fault tree and its BDD. 109

Figure 6.2 – Diagram showing the Event Node representation of a BDD *ite* node whose variable belongs to component *C* in phase *i*. 114
Figure 6.3 – Three phased mission for system with non-repairable component *A* and repairable component *B*. 118
Figure 6.4 - a) BDD from Wang and Trivedi method. b) BDD from method optimised for systems containing both repairable and non-repairable components. 118
Figure 6.5 – BDD representations of basic events for non-repairable and repairable components. 120
Figure 6.6 – BDD for phased mission fault tree from Figure 4.10 constructed through method given in section 6.4.2.1. 122
Figure 6.7 – Example phased mission for which method given mission unreliability upper bound. 124
Figure 6.8 – BDD for phased mission fault tree in Figure 6.7. 125
Figure 6.9 – Example phased mission for which method gives exact mission unreliability. 125
Figure 6.10 – Example phased mission for which method gives exact mission unreliability. 125
Figure 6.11 – Markov model for the repairable BDD on the right in Figure 6.6. 127
Figure 6.12 – Part of Implicant Tree data structure from BDD shown in Figure 6.6. 129
Figure 7.1 - Example System Fault Tree 133
Figure 7.2 – Phase fault trees for example phased missions. 155
Figure 7.3 - Example of 'cause' and 'effect' periods for phased mission importance measures. 160
Figure 8.1 - A simple three phased mission. 182
Figure 8.2 – Example phased mission used to demonstrate the improved decision making abilities of the Phased Birnbaum II importance measure. 184
Figure 8.3 – Another example phased mission used to demonstrate the improved decision making abilities of the Phased Birnbaum II importance measure. 185
Figure 8.4 – Safety system phased mission fault trees. 187
Figure 8.5 – Example phased mission for a distillation process. 195
Figure 8.6 – Fault trees for pump control system mission. 196
Figure 8.7 – Phase fault trees for the rocket launcher system. 199
Figure 8.8 – Plot of hazard rate against time for components in the rocket launching system. 199
Figure 8.9 – Phase fault tree for aircraft landing gear actuator system example. 206
Figure 8.10 – Plot of the change in the expected overall reliability cost of a phased mission against percentage reduction in phased mission failure probability for each component. 217

List of Tables

Table 2.1 - Common fault tree symbols 13
Table 2.2 - The t_1 and t_2 times for calculating probabilities of system failure in various time periods of a phased mission. 26
Table 3.1 - Component failure rates. 39
Table 3.2 – Component phase failure probabilities. 39
Table 3.3 - BDD node probability evaluation cases corresponding to Equations 3.79 to 3.82. 51
Table 4.1 – Rules for computing nodes *U* and *V*. 59
Table 4.2 – Parameters for supported component failure models. 73
Table 4.3 – Boolean AND operator truth table for BDDNode instance inputs. 81
Table 4.4 – Boolean OR operator truth table for BDDNode instance inputs. 82
Table 4.5 – The settings used in the random generation of 10 phased missions. 90
Table 4.6 - Average solution time comparison between the new and TD methods for each of the configuration settings. 91
Table 4.7 - Average BDD node count comparison between the new and TD methods for each of the configuration settings. 92
Table 4.8 - Average percentage error in the solution given by the TD method for each of the mission configuration settings. 93
Table 6.1 - Failure and repair rates for the components in the phased mission shown in Figure 6.7, Figure 6.9 and Figure 6.10. 124
Table 6.2 – System states for the repairable components from the BDD on the right in Figure 6.6. 126
Table 6.3 – State indexes for each system state from Table 6.2 in the state transition matrices for phases 1 and 2 of the mission. 127
Table 7.1 - Example System Failure Probabilities. 133
Table 7.2 – Probabilities for each mutually exclusive event combination in the example system. 134
Table 7.3 – Event combinations that contribute to RAW for each component in the example system. 136
Table 7.4 - Event combinations that contribute to RRW for each component in the example system. 137
Table 7.5 - Event combinations that contribute to Birnbaum importance for each component in the example system. 139
Table 7.6 – Event combinations that contribute to Criticality importance for each component in the example system. 141
Table 7.7 - Event combinations that contribute to Fussell-Vesely importance for each component in the example system. 143
Table 7.8 - Relationships between importance measures. 145
Table 7.9 - Event combinations that contribute to Group DIM importance for each component in the example system. 147
Table 7.10 - Event combinations that contribute to the Group RAW for each component in the example system. 151
Table 7.11 - Event combinations that contribute to Group RRW for each component in the example system. 153

Table 7.12 – Component phase failure probabilities. 156
Table 7.13 – Probabilities for each mutually exclusive event combination in the example mission. 156
Table 7.14 – Event combinations that contribute to the phase 1 in-phase ‘effect’ period Birnbaum importance measure value of each component in example missions *A* and *B*. 162
Table 7.15 - Event combinations that contribute to the in-phase ‘effect’ period Birnbaum importance for each component in example mission *A*. 163
Table 7.16 - Event combinations that contribute to the in-phase ‘effect’ period Birnbaum importance for each component in example mission *B*. 164
Table 7.17 - Event combinations that contribute to the value of the phase 2 transition ‘effect’ period Birnbaum importance measure with a phase 1 ‘cause’ period for each component in example mission *A*. 165
Table 7.18 - Event combinations that contribute to the value of the phase 2 transition ‘effect’ period Birnbaum importance with a phase 1 ‘cause’ period for each component in example mission *B*. 166
Table 7.19 - Event combinations that contribute to the in-phase phase 1 ‘effect’ period Criticality importance measure value for each component in example missions *A* and *B*. 168
Table 7.20 - Event combinations that contribute to the in-phase phase 2 ‘effect’ period Criticality importance measure value of each component in example mission *A*. 169
Table 7.21 - Event combinations that contribute to the in-phase phase 2 ‘effect’ period Criticality importance measure value of each component in example mission *B*. 170
Table 7.22 - Event combinations that contribute to the phase 2 transition ‘effect’ period Criticality importance measure value of each component in example mission *A*. 172
Table 7.23 - Event combinations that contribute to the phase 2 transition ‘effect’ period Criticality importance measure value of each component in example mission *B*. 173
Table 7.24 - Event combinations that contribute to the Mission Criticality importance measure value for each component in example mission *A*. 174
Table 7.25 - Event combinations that contribute to the Mission Criticality importance measure value for each component in example mission *B*. 175
Table 8.1- Comparison between the development of importance measures for non-phased mission and phased mission systems. 177
Table 8.2 – Effect of failure of a component in phase 1 on the probability of system failure in phase 2 for the example system described in Figure 8.1. 183
Table 8.3 - The importance measure values measuring the influence of phase 1 failure for each of the components in the phased mission shown in Figure 8.2. 184
Table 8.4 – The Phased Birnbaum and Phased Birnbaum II importance measure values for a phase 1 ‘cause’ period for each of the components in the phased mission shown in Figure 8.3. 185
Table 8.5 – Safety system component phase failure probabilities. 187
Table 8.6 – The phase 2 and phase 3 ‘effect’ period Phased Birnbaum II importance measure values for phase 1 ‘cause’ period for each component. 187
Table 8.7 – Component phase failure probabilities for example phased mission. 190
Table 8.8 – Mission Phased Criticality importance measure values assigned to each component by the two Criticality importance measure definitions for phased missions. 190

Table 8.9 - Component failure probabilities for each phase of pump control system mission. 196
Table 8.10 - Phase 2 Phased Fussell-Vesely importance for each component in the pump control system. 197
Table 8.11 – Cost of system failure during each phase of the phased mission. 214
Table 8.12 – One-off costs of reducing probability of component failure over the phased mission. 214
Table 8.13 – Direct cost of failure for each component in the phased mission. 215
Table 8.14 – Change in the expected phased mission reliability costs for different percentage reductions in the phased mission failure probability of component *A*..... 216

Nomenclature

A (as basic event).	Boolean variable representing the basic event for failure of component *A*.
A_b (as basic event).	Boolean variable representing the basic event for failure mode *b* of component *A*.
$A(x_i, t)$	Availability at time *t* for an item which has its failure event represented by x_i.
C_{a_i}	Cost incurred reducing the probability of component *i* failing.
$C_{d-after_i}$	Direct cost of component *i* failure after reducing its failure probability.
C_{d_i}	Direct cost of component *i* failure.
$C_{d-prior_i}$	Direct cost of component *i* failure prior to reducing its failure probability.
$C_{k,j}$	*k*th minimal cut set from the fault tree for phase *j* of a phased mission.
C_{sys}	Cost of system failure during a phased mission.
C_{total}	Total reliability cost of a phased mission.
$DIM^{II}{}^{x}_{\substack{i,j \\ m}}$	Period *x* 'effect' period Phased DIM^{II} importance measure with phase *m* 'cause' period for components *i* and *j*.
$DIM^{II}{}^{x}_{i,j}$	Overall period *x* 'effect' period Phased DIM^{II} importance measure for components *i* and *j*.
$DIM^{x}_{\substack{i,j,k \\ m}}$	Period *x* 'effect' period Phased Group DIM importance measure with a phase *m* 'cause' period for components *i*, *j* and *k*.

$DIM_{i_k}^{x}$	Period *x* 'effect' period Phased DIM importance measure with phase *k* 'cause' period for component *i*.
$DIM_{i,j,\dots,k}$	Group differential importance measure for component group $i, j, \dots, k$.
$DIM_{i,j,k}^{x}$	Overall period *x* 'effect' period Phased Group DIM importance measure for components *i*, *j* and *k*.
$DIM_{i,j}^{II}$	DIM^{II} importance measure for components *i* and *j*.
DIM_i	Differential importance measure for component *i*.
DIM_i^x	Overall period *x* 'effect' period Phased DIM importance measure for component *i*.
$D_i^{(C)}$	Matrix where the element at row *j* and column *k* is the probability that component *C* ends phase *i* in state *k* given that it started the phase in state *j* and failed at some point during the phase.
$E_i^{(C)}$	Matrix where the element at row *j* and column *k* is the probability that component *C* is in state *k* at the end of phase *i* given that it started the phase in state *j*.
$\mathrm{E}(x)$	Expected value of *x*.
FV_i	Fussell-Vesely importance measure for component *i*.
FV_i^x	Period *x* 'effect' period Phased Fussell-Vesely importance measure for component *i*.
$IB_{i_{j,k}}^{T}$	Phase *j* 'effect' period Transitional Birnbaum importance measure with phase *k* 'cause' period for component *i*.
$IB_{i_j}^{P}$	Phase *j* 'effect' period In-Phase Birnbaum importance measure for component *i*.
$IB2_{i_k}^{x}$	Period *x* 'effect' period Phased Birnbaum II importance measure with phase *k* 'cause' period for component *i*.
IB_i	Birnbaum importance measure for component *i*.

$IB_i(q_j = x)$	Conditional Birnbaum importance measure for component *i* where component *j* is in state *x*.
$IC_{i_j}^{P}$	Phase *j* 'effect' period In-Phase Criticality importance measure for component *i*.
$IC_{i_j}^{T}$	Phase *j* 'effect' period Transitional Criticality importance measure for component *i*.
$IC2_{i_k}^{x}$	Period *x* 'effect' period Phased Criticality II importance measure with phase *k* 'cause' period for component *i*.
$IC2_{i}^{x}$	Overall period *x* 'effect' period Phased Criticality II importance measure for component *i*.
$IC3_{\substack{i,j,k \\ t_1,t_2}}^{x}$	Period *x* 'effect' period Phased Group Criticality III importance measure with 'cause' period t_1 to t_2 for components *i*, *j* and *k*.
$IC3_{i_{t_1,t_2}}^{x}$	Period *x* 'effect' period Phased Criticality III importance measure with 'cause' period t_1 to t_2 for component *i*.
IC_i	Criticality importance measure for component *i*.
IC_i^M	Phased Mission Criticality importance measure for component *i*.
$JIB2_{\substack{i,j \\ m}}^{x}$	Period *x* 'effect' period Phased Joint Birnbaum II importance measure with phase *m* 'cause' period for components *i* and *j*.
$JRI_{i,j}$	Joint Reliability importance measure for components *i* and *j*.
N	Total number of phases in a phased mission.
N_{MCS_j}	Total number of minimal cut sets for phase *j* of a phased mission.
N_{MPS_j}	Total number of minimal path sets for phase *j* of a phased mission.

N_{WS_k}	Total number of working states in phase *k* of a phased mission.
N_{fm_i}	Total number of failure modes for component *i*.
N_C	Total number of components in system.
N_{mcs}	Total number of minimal cut-sets.
N_{ms}	Total number of Markov states to represent every possible set of states for the components in a system.
$P^{(C)}$	Transition rate matrix $Q^{(C)}$ but with no transitions out of states that correspond to failure for component *C* (i.e. no repairs).
P_{CEC_i}	The sum of the component event combination probabilities that contribute to importance measure *i*.
PFC_j	Phase failure combinations for phase *j* (intersection of the phase failure conditions for phase *j* and phase success conditions for subsequent phases).
$P_{k,i}(t)$	*i*th element of $P_k(t)$.
$P_{k,j}$	*k*th minimal path set from the fault tree for phase *j* of a phased mission.
$P_k(t)$	Probability vector for phase *k* where the *i*th element represents the probability that the system resides in state *i* at time *t* after the phase start.
$P(x_i)$	Probability that $x_i = 1$.
Q^{T_1,T_2}	Probability of system failure between times T_1 and T_2 of a phased mission. For non-repairable systems, this also corresponds with the system unreliability between t_1 and t_2 of a phased mission.

$Q^{(C)}$	Transition rate matrix where the element at row *i* and column *j* is the transition rate from state *i* to state *j* of component *C*.
Q_{Miss}	Probability of system failure at any time during a phased mission.
$Q_{Miss-INEX}$	INEX approximation of system failure at any time during a non-repairable phased mission.
$Q_{Miss-INEX-CC}$	INEX approximation of system failure at any time during a non-repairable phased mission with cut set cancellation applied.
$Q_{Miss-MCB}$	MCB approximation of the unreliability of a non-repairable phased mission.
$Q_{Miss-MCB-CC}$	MCB approximation of the unreliability of a non-repairable phased mission with cut set cancellation applied.
Q_{SYS}	Probability that system fails (non-phased mission).
Q_j	Probability of system failure in phase *j* of a phased mission.
Q_{j-INEX}	INEX approximation of the probability of system failure in phase *j* of a non-repairable phased mission.
Q_j^P	Probability of in-phase system failure in phase *j* of a phased mission.
Q_j^T	Probability of system failure on transition to phase *j* of a phased mission.
$Q(x_i, t)$	Unavailability at time *t* for an item which has its failure event represented by x_i.
R_{Miss}	Probability of a phased mission completing successfully (i.e. mission reliability).
R_{j-MCB}	MCB approximation of the reliability of phase *j* in a non-repairable phased mission.

$R(x_i, t)$	Reliability at time *t* for an item which has its failure event represented by x_i.
$SUP_{k,j}$	Set of cut sets from phases 1 to *j-1* that are supersets (using definition of superset by Dazhi and Xiaozhong [1]) of the *k*th cut set from phase *j*.
S_j	Success conditions (union of path sets) for phase *j* of a phased mission.
T_i	Earliest time at which $x_i = 1$.
$U(x_i, t)$	Unreliability at time *t* for an item which has its failure event represented by x_i.
X_j	Failure conditions (union of cut sets) for phase *j* of a phased mission.
$X_{m,i}^{(C)}$	Matrix where the element at row *p* and column *q* is the probability that component *C* ends phase *i* in state *q* given that it ended phase *m* in state *p* and the component fails in phase *i*.
$Y_{n,i}^{(C)}$	Matrix where the element at row *p* and column *q* is the probability that component *C* ends phase *i* in state *q* given that it ended phase *n* in state *p* and the component does not fail in phase *i*.
$a_{i,j,k}$	Group RAW importance measure for components *i*, *j* and *k*.
$a_{i,j,k}^{x}$	Period *x* 'effect' period Phased Group RAW for components *i*, *j* and *k*.
a_i	Risk Achievement Worth (RAW) importance measure for component *i*.
a_i^x	Period *x* 'effect' period Phased RAW importance measure for component *i*.
$b_i^{(C)}$	Boolean variable representing the state of component *C* during phase *i*. Equals 1 if the component fails during phase *i* and 0 otherwise.

$cp(x_i)$	Component of the event represented by the Boolean variable x_i.
$f_{x_i=0}$	Boolean function where variable x_i is set equal to 0.
$f_{x_i=1}$	Boolean function where variable x_i is set equal to 1.
$fm(x_i)$	Failure mode of the event represented by the Boolean variable x_i.
$index(x_i)$	Index of x_i where this is defined by a global ordering across all Boolean variables.
$\inf(s)$	Infimum of *s* (lowest number in *s*, if *s* is a set of real numbers).
$k_{k,j}$	Total number of basic events in the *k*th cut set from the fault tree representing system failure in phase *j* of a phased mission.
$pn(x_i)$	Phase of the event represented by the Boolean variable x_i.
q_{i_k}	Probability that component *i* fails in phase *k*.
q_i	Probability that component *i* fails.
$q_i^-(t, t_1, t_2)$	Probability that component *i* is failed at time *t* if its probability of failure between times t_1 and t_2 is reduced by 1% over its baseline (actual) value.
$r_{i,j,k}$	Group RRW importance measure for components *i*, *j* and *k*.
$r_{i,j,k}^x$	Period *x* 'effect' period Phased Group RRW for components *i*, *j* and *k*.
r_i	Risk Reduction Worth (RRW) importance measure for component *i*.

r_i^x	Period *x* 'effect' period Phased RRW importance measure for component *i*.
$r_{k,j}$	Probability of the occurrence of cut set *k* from the fault tree for phase *j*.
$\sup(s)$	Supremum of *s* (greatest number in *s*, if *s* is a set of real numbers).
t_0	The start time of a phased mission.
t_j	The time at which phase *j* of a phased mission ends.
t_j^-	$t_j - dt$, i.e. the time just before phase change at the end of phase *j*.
t_j^+	$t_j + dt$, i.e. the time just after the phase change at the end of phase *j*.
$v_0^{(C)}$	Initial probability vector for component *C*, where the *i*th element is the probability that the component is in state *i* at the start of the phased mission.
$w^{(x)}$	Probability vector for a BDD node, G, with variable *x* belonging to component *C* in phase *i*. The *k*th element is the probability that both *G=1* and component *x* is in state *k* at the end of phase *i*.
x_i	Boolean variable representing the state of event *i*. $x_i = 1$ if the event has occurred, $x_i = 0$ otherwise.
$\overline{x_i}$	The complement of x_i. If $x_i = 1$ then $\overline{x_i} = 0$ and vice versa.
$x_i(j,\infty)$	Boolean variable representing the occurrence of event *i* between times t_j and ∞, i.e. after t_j .
$x_i(j,k)$	Boolean variable representing the occurrence of event *i* between times t_j and t_k .

$x_i(t)$	x_i at time *t*.
β	Transition rate matrix where the element at row *i* and column *j* is the transition rate from state *j* to state *i* of the system.
$\beta(i,j)$ and $\beta_k(i,j)$	Element at row *i* and column *j* of the transition rate matrix.
β_k	Transition rate matrix for phase *k* of a phased mission.
$\delta_k(i)$	The index of state *i* from the reduced state probability vector in the full probability vector for phase *k*.
$\lambda_{j,i}$	Transition rate from state *j* to state *i*.
$\rho_k(i)$	The index of state *i* in the reduced state probability vector (containing only working states) for phase *k*.
σ_j	Boolean variable that equals 1 during phase *j* and 0 otherwise.
$\psi_i(t)$	Failure probability density for component *i* at time *t*.
$\phi_{k,j}$	Boolean variable that is equal to 1 if state *j* is a failure state during phase *k* and equal to 0 otherwise.
$\oplus$	Boolean logical operator (e.g. AND or OR).
∧	Boolean AND operator.
∨	Boolean OR operator.

Acronyms

ALARP	As Low as Reasonably Practicable
BDD	Binary Decision Diagram
DIM	Differential Importance Measure
EZ Method	Esary and Ziehms Method
GUI	Graphical User Interface
Ite Structure	If-then-else Structure
JRI	Joint Reliability Importance Measure
MFOP	Maintenance Free Operating Period
PDO	Phase Dependent Operation
RAW	Risk Achievement Worth
RBD	Reliability Block Diagram
RRW	Risk Reduction Worth
TD Method	Tang and Dugan Method
UAV	Unmanned Aerial Vehicle

1 Introduction

A system that performs a series of tasks that are carried out over consecutive time periods to meet some overall goal is known as a phased mission system, where each period is known as a phase. Ensuring that they perform with high reliability is often critical due to the importance of achieving their mission goals and avoiding the consequences of failure, but is often difficult to achieve since the system must perform without failure in every phase.

To design a phased mission system with a desired reliability level requires the ability to:

1. Predict the reliability of a particular system design.
2. Identify the areas of a system design that should be modified to approach the desired reliability.

If those two requirements are fulfilled then the design of a highly reliable phased mission system can be accomplished, given the availability of sufficient resources, by following the strategy shown in Figure 1.1.

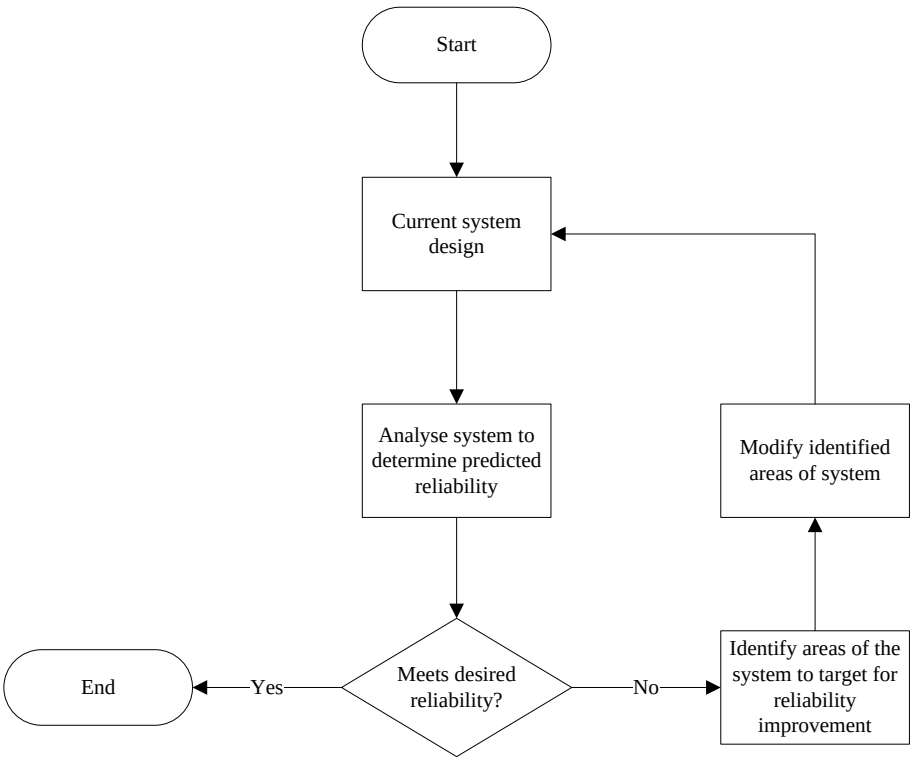


Figure 1.1 – Strategy for designing a high reliability phased mission system.

However, it is well known that the analysis of phased mission system reliability is far more complex than the non-phased mission case due to the system reliability structure variation between phases and the presence of dependencies between component failures in different phases [1]. The overall complexity of a reliability analysis of a phased mission system is dependent on a number of factors such as the dependencies present, e.g. those due to repair and components with multiple failure modes, the complexity of the reliability structure of the

system, e.g. number of components and presence of redundancy, and the complexity of the phased mission itself, e.g. the number of phases involved. This is illustrated by Figure 1.2.

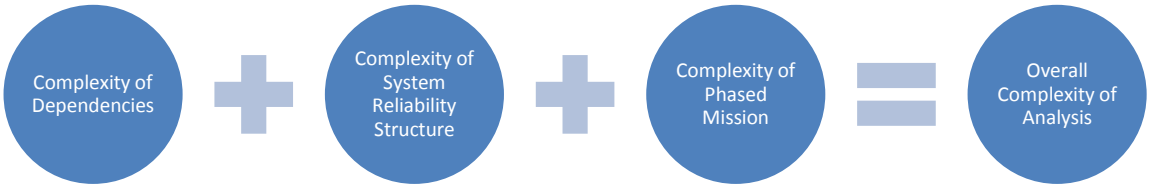


Figure 1.2 - Factors influencing the complexity of analysing phased mission system reliability.

The ability to analyse the reliability of a phased mission depends not only on the complexity of the analysis but also on the computational resources and time that are available and the efficiency of the analysis method used, as shown in Figure 1.3.

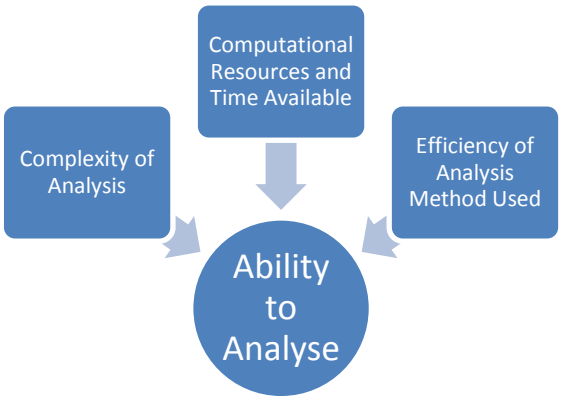


Figure 1.3 - Factors affecting ability to analyse the reliability of a phased mission system.

Unfortunately, the phased mission systems encountered in the real world are usually of the high complexity type and simplifying assumptions cannot usually be made without adversely affecting the accuracy of the results. The computational resources and time available are also usually limited due to the constraints on budgets and short design timeframes that are commonly present in industry. Thus, the practical analysis of the reliability of real world phased mission systems is dependent on the development of efficient methods.

For identification of areas of a system that should be targeted for reliability improvement in order to reach some system reliability goal, the use of importance measures are widely used in the analysis of non-phased mission systems. These show the most important components in the system, for example those currently contributing to

a large proportion of system failures or those for which a given reliability improvement has the greatest affect on the reliability of the system. However, those developed for non-phased mission systems cannot be used for the analysis of those operating in phased missions and only recently have importance measures specific to phased missions been developed [2]. The systematic and convenient optimisation of the reliability of phased mission systems relies on the further development of these and new importance measures.

This thesis presents methods for the prediction of phased mission system reliability with greater efficiency and the precise identification of areas of a system to be targeted for reliability improvement to reach various system reliability goals. They therefore provide improved means to perform both of the important analysis steps from the strategy for designing highly reliable real world phased mission systems that was outlined in Figure 1.1.

1.1 Common Properties of Real World Phased Mission Systems

The work has been developed with the aim of improving the methods available for the reliability analysis of the types of real world phased mission system that have the following in common:

- They are constructed from the integration of large numbers of components to form a complex system.
- The combinations of component failures that cause system failure vary between phases.
- The failure of the system during any phase results in immediate failure of the mission.
- There are consequences if the system is unable to carry out the mission and these may depend on the phase in which failure occurs.

A typical example of such a system is a retractable aircraft landing gear system, such as that fitted to the Boeing 777-300 aircraft and shown in Figure 1.4, for which each flight can be considered a mission consisting of the following phases:

1. Taxi
2. Take-Off
3. Climb
4. Cruise
5. Descent
6. Landing
7. Taxi

This type of landing gear system is extremely complex and consists of a huge array of different parts, ranging from mechanical actuators to electronic sensors.



Figure 1.4 – Photo of a retractable landing gear system from a Boeing 777-300 aircraft.

It performs different roles in each phase resulting in a variation in the combinations of component failures that cause system failure. During taxi it supports the weight of the aircraft, during climb it stows itself away within the fuselage and during landing it absorbs the force of the impact with the runway. For example, a failure of the hydraulic system that powers the retraction and extraction of the landing gear would only cause mission failure during climb and descent, the phases in which that subsystem is used. The failure of the system in any phase would lead to the ending of the mission, the flight may have to be aborted, an emergency landing may need to be made, such as in the case of the JetBlue Airways aircraft (Flight 292) that is shown in Figure 1.5, and at the very least, the aircraft will not be able to fly again until an investigation into the cause of the failure and a repair of the system has been made.



Figure 1.5 – Photo of a JetBlue Airways aircraft making an emergency landing after the front landing gear malfunctioned during take-off.

Finally, there are consequences to the system not completing a mission successfully. The precise consequences may depend on the phase in which the failure occurs, for example the consequences of failure during taxi will generally be mild in comparison to a failure during landing, and range from the catastrophic loss of the aircraft to economic losses due to flight cancellations and delays.

1.2 Reliability Analysis of Phased Missions

Due to the need to meet the mission goals and avoid the consequences of failure, the ability of these real world systems to carry out their missions with high reliably, where reliability is defined as the probability that it is completed successfully, is often critical. Creating systems that attain high reliability is nonetheless difficult due to the sophistication of modern engineering systems and the complexity of the phased missions that they undertake. The designers, manufacturers and operators therefore need to understand how to achieve high reliability with new systems and require assurance and insight into the improvement of the reliability for those that already exist. One solution is to over-engineer such systems, perhaps by incorporating great levels of redundancy or extremely robust components, in the hope of gaining high reliability. However this approach can be detrimental to other aspects of the design such as leading to higher costs and is also still not guaranteed to result in the desired reliability. A better approach is to apply methods from reliability engineering to create a model of the system from the known logical connections between the failed status of the system and its components, together with the probabilistic models for failure and repair of the latter, so that the reliability of the system design can be accurately predicted and optimised in a rigorous manner. Some of the fundamental definitions, theories and methods from reliability engineering are presented in chapter 2 of this thesis, many of which form the foundation for the methods presented in later chapters.

1.2.1 Reliability Measurement

The measurement of the mission reliability and phase failure probability for a system operating in a phased mission is an important metric. However, the introduction of statistical time dependencies between components in different phases complicates the analysis significantly compared to the non-phased case. In the 35 years since Esary and Ziehms [3] first addressed this problem there have been two main approaches to the analysis of phased mission reliability:

- Combinatorial such as the fault tree and Binary Decision Diagram.
- State space such as the Markov method and Petri-nets.

The combinatorial methods are generally far more efficient computationally but are unable to deal with the statistical dependencies introduced by repair and are therefore used in the analysis of non-repairable systems. State space methods are able to deal with these dependencies but suffer from what is known as the state-space explosion problem, meaning that the analysis of large systems or those that operate over many phases becomes intractable.

Due to the complexity of phased mission reliability analysis and the large number of components contained within modern systems, any real analysis will need to be performed by computer. Despite the advancements made, the sizes of the phased mission problems that can be analysed on a modern commodity computer are still very limited, particularly for repairable systems. In addition, certain use cases require the performance of repeated reliability analysis in less time and with less memory, such as unmanned aerial vehicles (UAVs) optimising the phase configuration of their mission based on real time reliability predictions. There is therefore a

compelling need for the development of phased mission reliability measurement methods that have improved computational efficiency.

Chapter 3 of this thesis discusses the existing methods available for the analysis of phased mission reliability of non-repairable systems that were found in the literature. Many of these utilise some form of Boolean algebra to deal with the phase dependencies between component failures and the methods presented include both the early fault tree based approaches and the more recent, and much improved, analysis methods that use the Binary Decision Diagram (BDD) technique. Chapter 4 highlights some of the problems and areas for improvement that were discovered during the study of these methods. These include the inaccuracies found in the method for analysing systems that contain multiple failure mode components and the lack of a method optimised for the repeat reliability analysis of a system design that occurs when performing a reliability improvement exercise or real time analysis. A newly developed method that addresses both of these problems is then presented and shown to be an improvement. Moving to repairable phased missions systems, chapter 5 presents the methods found in the literature for their reliability analysis and these include methods utilising the Markov, BDD and Petri net techniques. Each of these methods has its own weaknesses, for example the Markov methods are unable to analyse the reliability of very large or complex phased mission systems due to the state space explosion problem and the BDD technique is limited to the case where repaired components are only integrated back into the system at the end of the phase in which they are repaired. Five new methods for the analysis of repairable phased mission systems are presented in chapter 6, some of which are extensions to the existing methods and others which are entirely new. These include extending an existing Markov method to allow it to analyse systems containing components with multiple failure modes, a new BDD method for identifying which system states represent system failure, and two new methods that are more efficient in the analysis of systems that contain both repairable and non-repairable components – a common case in real world systems.

1.2.2 Reliability Improvement

There are two ways in which the reliability of a phased mission system can be improved. The first is to alter its phase reliability structure, for example by adding additional redundancy. The second is to alter the reliability of the components used, perhaps by using more robust alternatives or increasing preventive maintenance. In either case, assuming a limited budget, a choice must be made as to which part of the system and which components should be targeted. This is a demanding task if the system is complicated, contains many components or the phased mission is large. Importance measures, which give an index of the influence or contribution of a component or group of components on the system reliability, can be used to help with this type of decision and also to predict the magnitude of improvement that can be obtained. For non-phased mission systems there are six commonly used importance measures available, namely the Birnbaum, Criticality, Fussell-Vesely, Risk Reduction Worth, Risk Achievment Worth and Differential importance measures. Each has a unique interpretation of component importance, and each appropriate for use with certain reliability improvement goals. For example, the Birnbaum importance measure shows those components with highest structural importance and therefore those for which a fixed reliability improvement would lead to the largest reduction in system reliability whilst the Risk Reduction Worth importance measure highlights those that provide the scope for achieving the greatest improvement in system reliability. Importance measures can provide even more precise

data for system reliability improvement with systems operating in phased missions. For example, they can show how the reliability of the system can be improved specifically in the phase of the mission with the highest consequence of failure. However, whilst the development of importance measures for non-phased mission systems has reached an advanced state, significantly less research into phased mission importance measures has been carried out. The further development of those suitable for phased mission systems is therefore required since importance measures have an important role in facilitating reliability optimisation.

Chapter 7 is a review of the existing importance measures and covers those that have been developed for both non-phased and phased mission systems. The importance measures range from the widely used and well known Birnbaum importance measure to the recently introduced Differential importance measure, and include both those for measuring the importance of a component and groups of components. Notable is that significantly less importance measures have been developed for phased mission systems and that no method for dealing with the cost of system failure, which may vary dependant on the phase in which the system has failed, has been presented. Chapter 8 seeks to address this imbalance by presenting several new importance measures that have been developed to help with the optimisation of phased mission system reliability. These include interpretations of many of those previously only available for non-phased mission systems, those with the ability to find the importance of components with respect to precise time periods and the first developed for the measurement of the importance of a group of components.

1.3 Research Objectives

The objectives of the research presented in this thesis are to provide the following for phased mission systems:

1. Computationally efficient methods for the measurement of phase and mission reliability of both non-repairable and repairable systems.
2. Importance measures that provide information that can be used to direct and predict the outcome of a variety of phased mission system reliability improvement and optimisation actions.

2 Reliability Theory and Definitions

2.1 Introduction

The United States Defence Standard [4] defines reliability as "The probability that an item will perform a required function without failure under stated conditions for a stated period of time". The reliability of systems is a topic of great importance to both business and society in general. Increasing competition, safety legislation and customer expectation levels is pushing the search for ever higher levels of reliability in a wide range of systems ranging from military aircraft to everyday household appliances. The consequences of system failure, the failure of a system to meet its intended objectives, vary from the catastrophic, such as the Chernobyl nuclear power plant disaster [5] in 1986 which caused multiple fatalities, to financial, such as the fines incurred by UK electricity suppliers when customers suffer a loss of supply [6]. The field of reliability and risk assessment advances theory, methods and tools that lead to enhanced understanding and improvement of the reliability of systems.

This chapter gives a brief overview of theory and definitions that are commonly used in this field.

2.2 Definition of Risk

Risk can be defined as the consequence, C, of a hazardous event occurring multiplied by the probability or frequency of the event's occurrence, P, as shown by Equation 2.1. The risk can therefore be decreased by reducing the consequence of failure or the frequency of failure.

$$Risk = C \times P \qquad \textbf{2.1}$$

The consequence can be measured in a number of ways depending on the analysis being carried out. For example, it could be defined as the number of fatalities caused by exposure to radiation from a nuclear power station safety system failure or as the number of tonnes of oil spilled by an oil tanker. The acceptable level of risk for some application can be determined through a cost benefit analysis and is often subject to the law of diminishing returns [7] such that the reduction in risk for a given input of resources decreases as the level of risk is decreased. For safety application, the acceptance means of the UK Health and Safety Executive is "as low as reasonably practicable" (ALARP) [8], which means that, in order to comply with UK health and safety law, duty-holders should carry out risk reduction measures unless the costs are deemed to be grossly disproportionate to the level of risk reduction that would be achieved.

2.3 Components and System Definitions

Components are generally defined as the items at the 'limit of resolution of the system', items whose reliability is not described in terms of their constituent parts. This may be a subsystem, consisting physically of more than one part, but which is treated as a single entity with its own statistical reliability model for the purposes of the analysis being carried out. A component may have multiple failure modes, meaning that it can fail in more than

one distinct way. Usually these failure modes cannot occur simultaneously and such failure modes are mutually exclusive. A component failure mode is usually denoted by the component name with the failure mode name as a subscript, e.g. A_b denotes failure mode *b* of component *A*. An example of a component with multiple mutually exclusive failure modes is a switch whose failure modes include sticking in the closed and open positions.

A system is a set of components that interact to perform some overall function. Determining what to include in the definition of the system for the purposes of a reliability analysis is an important step. External systems and other factors that influence the system but are not part of the system itself are deemed external to the 'system boundary' and excluded from the analysis.

2.4 Reliability Metrics

Reliability metrics provide useful information on the frequency and probability of failure of the system. These metrics are used to present the current or predicted level of system reliability, compare the reliability between systems and to set reliability targets. A high proportion of reliability engineering focuses on methods to determine and predict the values of these metrics.

A few of the most commonly used metrics are described in this section.

2.4.1 Item State

Let $x_i(t)$ denote the state of event *i* (typically, the occurrence of an item failure where the item could be a component, subsystem or subsystem) at time *t*. So:

$$x_i(t) = \begin{cases} 1 & if\ event\ i\ has\ occured, \\ 0 & otherwise. \end{cases}$$

2.4.2 Time to first failure

If event x_i represents a failure event for an item then the time to first failure, T_i, is the time that elapses between it being new (and assumed working), *t*=0, to the time that it fails for the first time (i.e. earliest time at which $x_i = 1$).

2.4.3 Reliability and Unreliability

The unreliability of an item at time *t* which has its failure event is represented by x_i, $U(x_i, t)$, is defined as the probability that it fails one or more times before *t*, given that it was operating at *t* = 0. It is therefore represented by the cumulative distribution function shown in Equation 2.2.

$$U(x_i, t) = P(T_i \leq t | x_i(0) = 0) \qquad \textbf{2.2}$$

The reliability of an item at time *t* which has its failure event represented by x_i, $R(x_i, t)$, is the probability that it works continuously from *t* = 0 to time *t*, as shown by the complementary cumulative distribution function in Equation 2.3. It is a very important metric for failure critical systems.

$$R(x_i, t) = P(T_i > t | x_i(0) = 0) \quad \textbf{2.3}$$

The relationship between unreliability and reliability is shown in Equation 2.4.

$$U(x_i, t) = 1 - R(x_i, t) \quad \textbf{2.4}$$

2.4.4 Availability and Unavailability

The availability of an item at time *t* which has its failure event represented by x_i, $A(x_i, t)$, is defined as the probability that the item is working at time *t* as shown by Equation 2.5.. For a non-repairable system or component this is equivalent to the reliability.

$$A(x_i, t) = P(x_i(t) = 0) \quad \textbf{2.5}$$

The unavailability of an item at time *t* which has its failure event represented by x_i, $Q(x_i, t)$, is defined as the probability that the item is in a failed at time *t* as shown by Equation 2.6.

$$Q(x_i, t) = P(x_i(t) = 1) \quad \textbf{2.6}$$

The relationship between availability and unavailability is shown in Equation 2.7.

$$A(x_i, t) = 1 - Q(x_i, t) \quad \textbf{2.7}$$

2.5 System Reliability Structure Representation

Usually a system's reliability will be analysed through a combination of data on the reliability behaviour of its subsystems or components and the relationship between the reliability of these and the complete system. This is often far more practical than dealing with system level as directly obtaining data on the system reliability may take too long, be unsafe, be impractical or not even possible – for example when the system exists only as a design. Furthermore such analysis is far more useful as it can be used to discover the areas of the system contributing to failure, how the system reliability can be improved and used to determine the reliability of the system under different scenarios and conditions. This leads to the requirement for methods used to describe the

system reliability in terms of the reliability of its subsystems and components, known as the system reliability structure.

The system reliability structure can be represented by a Boolean function of Boolean variables, known as a system's structure function. Each Boolean variable represents the state of one of the system's components. If the structure function evaluates to one for a given mapping of component states, then this set of component states results in system failure, otherwise the function evaluates to zero and the system works.

Except for trivial systems, it is impractical to derive the structure function directly from a system description or understanding. Instead representations are used that can be produced from a system description. These communicate the system reliability logic graphically and allow fast quantitative and qualitative analysis. Several methods exist for the efficient representation of these structure functions in the form of directed acyclic graphs. The most widely known and used methods are event trees, reliability block diagrams, fault trees and binary decision diagrams. Each of these system reliability structure representations is discussed in this section.

2.5.1 Event Trees

An Event Tree [9] is a graphical representation in the form of a tree that begins with an initiating event (usually some deviation from normal system operation) and then considers further system events that could occur, following the possible paths to their final consequences. At each possible system event, a new node is added to the tree, branching into two each time with one branch representing the occurrence of the event and the other representing its non-occurrence. It is most commonly used to model safety systems and determine the likelihood of consequences of various severities after the occurrence of a hazardous event. An example of an event tree is shown in Figure 2.1.

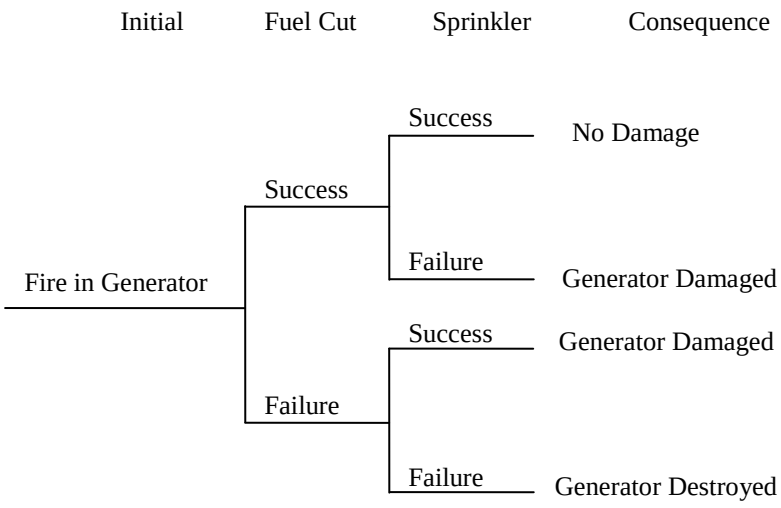


Figure 2.1 – An example of an event tree.

2.5.2 Reliability Block Diagrams

Reliability block diagrams (RBD) [10] describe the reliability structure of the system as a connected network of components that are required to maintain the system in an operational state. They consist of a start and an end node, a series of intermediate nodes, a series of edges each representing a component and an incidence function associating each edge with its adjacent pair of nodes. If a path from the start to the end node passing through working components (edges) exists then the system is in a working state. If a component fails then there is no path through that component. Figure 2.2 shows an example of an RBD; here the system fails if A fails or if B and C both fail together.

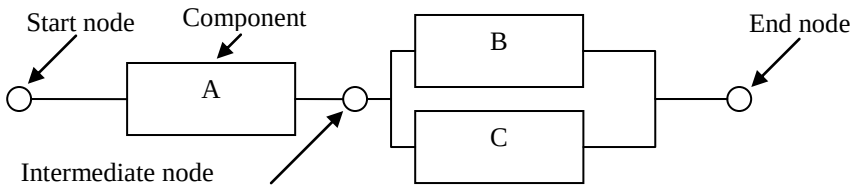


Figure 2.2 - A Simple Reliability Block Diagram

2.5.3 Fault Trees

Fault trees [11] are a graphical representation of the combination of basic event failures (component failure modes) that lead to the top event (system failure). The events are connected by Boolean logic gates and can be used for both qualitative and quantitative analysis. They are easy for people to understand and produce from knowledge of the system reliability logic.

An example of a simple fault tree, equivalent to the RBD in Figure 2.2, is shown in Figure 2.3.

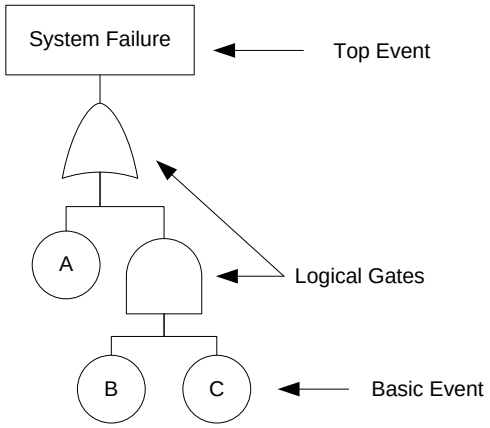
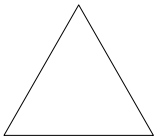


Figure 2.3 - A simple fault tree

The construction of a fault tree begins with an undesired state of the system, known as the top event, whose structure function is to be represented. The causes of this top event are deduced and connected to it via logical gates representing the logical relation between the causes and the top event. This process is continued with each successive cause broken down into its causes, until the limit of resolution is reached. This limit of resolution is usually the component level, for which reliability data exists thus allowing the system reliability to be calculated. The lowest levels of the fault tree therefore consist of basic events.

The three most commonly used gate symbols correspond to Boolean logical operations; an AND gate to the 'union' operation, an OR gate to the 'intersection' operation and a NOT gate to the 'complementation' operation. A fault tree such that increased unreliability of any of its basic events cannot reduce the top event probability is known as coherent (meaning that the Boolean function that the fault tree represents is monotonically increasing). Thus, fault trees containing NOT gates can be non coherent [12] whilst those consisting of only AND and OR gates are always coherent. The most commonly used components of a fault tree are shown in Table 2.1.

Table 2.1 - Common fault tree symbols

Name of Gate	**Fault Tree Symbol**	**Description**
OR		The output occurs if at least one input occurs.
AND		The output occurs if all inputs occur.
NOT		The output occurs if the input doesn't occur.
Transfer		Represents a sub fault tree that is developed elsewhere.

2.5.4 Binary Decision Diagrams (BDDs)

2.5.4.1 Motivation

Although fault trees are convenient for people to understand and to produce from an understanding of the underlying system reliability logic, it is computationally expensive to derive the exact top event probability

directly from them. This is particularly the case for large fault trees, for which the solution can be difficult to compute within acceptable processing and memory constraints. Approximate solution methods for fault trees exist, such as the rare event approximation [13], but the error can be large – particularly if the tree is non-coherent since the usually high probability of success events falsifies the rare event assumption. A fault tree is therefore often converted into a BDD prior to analysis.

2.5.4.2 BDD Representation of a Boolean function

A binary decision diagram (BDD) [14] is a compact data structure, in the form of a rooted, directed, acyclic graph, which can be used to represent and manipulate the Boolean function representing a system's reliability. They were introduced by Bryant [15] and are based upon Shannon decomposition theory [16]. The compact nature results from two important reduction features:

1. The merging of isomorphic subgraphs
2. The elimination of any node whose children are isomorphic

A BDD consists of decision nodes and two terminal nodes called terminal 0 and terminal 1. Each decision node is labelled with a Boolean variable and has two edges, a 0 edge and 1 edge, each of which connects to a child node. The 0 edge represents an assignment of 0 to the node's Boolean variable, whilst the 1 edge represents an assignment of 1. The Boolean variables are ordered such that if an edge from a node labelled with variable i connects to a node labelled with variable j then $i < j$. The chosen variable ordering often has a significant impact on the size of the resultant BDD. Unfortunately finding the optimum variable ordering is an NP-hard problem [17], although heuristic techniques have been developed that enable good orderings to be obtained efficiently [18].

A BDD starts at a single node known as its top or root node. An example of a BDD is shown in Figure 2.4. The value of the Boolean function represented by the BDD corresponding to the mapping of Boolean values to variables on any route from the root node to a terminal node, is given by the terminal nodes value. These routes are known as the BDD's paths. Often not all variables will appear on a route through the BDD and these are variables whose value has no bearing on the Boolean function value, given the variable value assignments on the path. Such variables are sometimes known as 'doesn't matter variables'.

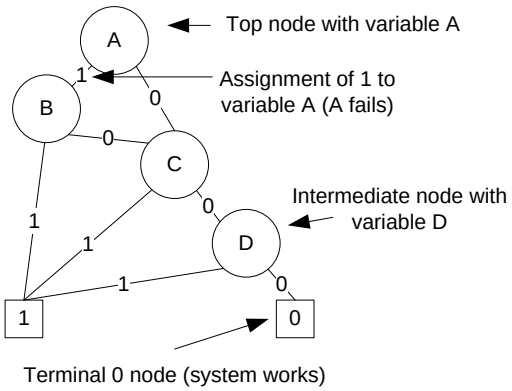


Figure 2.4 – An example of a BDD with variable ordering: A < B < C < D.

The BDD structure can be presented in terms of a series of nested i*f-then-else (ite)* structures, each representing a decision node in the BDD. The *ite* structure represents the decomposition of a Boolean function *f*, of Boolean variables $(x_1, x_2, \ldots, x_n)$, around x_i as given by Equation 2.8 where $f_{x_i=1}$ and $f_{x_i=0}$ are defined in Equation 2.9 and Equation 2.10 respectively.

$$f = (x_i = 1) \wedge f_{x_i=1} \vee (x_i = 0) \wedge f_{x_i=0} \quad \textbf{2.8}$$

$$f_{x_i=1} = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \quad \textbf{2.9}$$

$$f_{x_i=0} = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \quad \textbf{2.10}$$

Using the *ite* format this becomes Equation 2.11.

$$f = ite(x_i, f_{x_i=1}, f_{x_i=0}) \quad \textbf{2.11}$$

Equation 2.11 describes the following situation, if variable x_i occurs (fails) then $f_{x_i=1}$ is considered, else $f_{x_i=0}$ is considered.

2.5.4.3 Forming a BDD from a fault tree

It is not practical to directly derive a BDD from a system description due to the difficulty in forming the correct logical structure and the huge numbers of nodes and edges that they often contain. In comparison the fault tree representation is compact and intuitive. It is therefore normal to form a fault tree and then convert it into a BDD through the application of an algorithm.

To form the BDD representation of a fault tree, the first step is to convert each of the basic events in the fault tree into their *ite* structure representations.

Any basic event in the fault tree, for example x_i, maps to the *ite* structure shown in Equation 2.12.

$$x_i = ite(x_i, 1, 0) \quad \textbf{2.12}$$

To represent the success of a component the terminal nodes are switched. The *ite* form of the event that x_i doesn't occur is shown in Equation 2.13.

$$\overline{x_i} = ite(x_i, 0, 1) \quad \textbf{2.13}$$

The *ite* structure representation of a gate in the fault tree is formed by performing the appropriate logical operation on the *ite* structures representing the gate inputs. If the gate has more than two inputs, then the Boolean logical operation is first performed on the initial two inputs and the resultant *ite* structure obtained. The logical operation is then performed on this structure and the next gate input. This process of combining the resultant *ite* structure from previous gate inputs and the next gate input's *ite* structure is continued until all gate inputs have been processed and the final *ite* structure obtained.

A logical operation such as AND or OR between two nodes *F* and *G*, presented in *ite* form in Equations 2.14 and 2.15 respectively, outputs a new node as shown by Equation 2.16, where $\oplus$ represents a logical operation (AND or OR), $U = (F \oplus G)_{x=1} = F_{x=1} \oplus G_{x=1}$ and $V = (F \oplus G)_{x=0} = F_{x=0} \oplus G_{x=0}$.

$$F = ite(x, F_{x=1}, F_{x=0}) = ite(x, F1, F0) \quad \textbf{2.14}$$

$$G = ite(y, G_{y=1}, G_{y=0}) = ite(y, G1, G0) \quad \textbf{2.15}$$

$$F \oplus G = ite(x, U, V) \quad \textbf{2.16}$$

The failure child node, U, and success child node, V, of the node output by the operation are given by Equation 2.17 to Equation 2.18, where *index(x)* is an index given Boolean variable *x* from a global ordering across all Boolean variables.

$$U = \begin{cases} F1 \oplus G1, & if\ index(x) = index(y) \\ F1 \oplus G, & if\ index(x) < index(y) \end{cases} \quad \textbf{2.17}$$

$$V = \begin{cases} F0 \oplus G0, & if\ index(x) = index(y) \\ F0 \oplus G, & if\ index(x) < index(y) \end{cases} \quad \textbf{2.18}$$

The BDD is formed through the recursive application of Equation 2.16 whilst applying the following reduction rules:

- Identical sub trees are shared.
- Any node whose failure and success child is the same node is removed and replaced by that child node.

The conversion process for the example fault tree in Figure 2.5a that results in the BDD shown in Figure 2.5b, when the variable ordering is A < B < C, is:

$$ite(A,1,0) \wedge ite(B,1,0) \wedge ite(C,1,0) = ite(A, ite(B,1,0),0) \wedge ite(C,1,0) = ite(A, ite(B, ite(C,1,0),0),0)$$



Figure 2.5 - A fault tree and an equivalent BDD.

2.6 Representing the top event in terms of basic event combinations

A common form of qualitative analysis is to determine the combinations of basic events that cause system failure when they all occur. Explained in this section are cut sets and implicants, widely used basic event combinations with specific properties.

2.6.1 Cut Sets

A cut set is any combination, and a minimal cut set a least combination, of basic events that cause system failure when they occur. The difference is therefore that a subset of a cut sets basic events occurring may cause system failure whereas a minimal cut set requires that all basic events in the set occur. For example a minimal cut set consisting of basic events representing the failure of components A, B and D requires all three of the components to be failed for the system to fail. Minimal cut sets can be determined from a set of cut sets by applying the distributive, idempotent and absorption Boolean algebra rules demonstrated in the example equations 2.19, 2.20 and 2.21 respectively. The idempotent rule removes repeated cut sets and failure events whilst the absorption rule removes redundant cut sets.

$$(A \vee B) \wedge (C \vee D) = (A \wedge C) \vee (A \wedge D) \vee (B \wedge C) \vee (B \wedge D) \quad \textbf{2.19}$$

$$A \vee A = A \wedge A = A \quad \textbf{2.20}$$

$$A \vee (A \wedge B) = A \quad \textbf{2.21}$$

2.6.2 Implicants

An implicant is a similar concept to that of a cut set but includes success events as well as failure events. They are often derived from non-coherent fault trees or from BDDs.

2.7 Quantification methods

The methods used to perform quantitative analysis on fault trees and BDDs, such as determining the probability of failure of the system, are explained in this section.

2.7.1 Fault Tree Based Quantification Methods

To quantify the probability of the occurrence of a fault tree's top events, the minimal cut sets (or minimal implicant sets if the fault tree is non-coherent) are used. The 'Bottom-up' approach can be used to obtain the minimal cut sets from a fault tree. This approach begins with the basic events of the tree which are then expanded into higher level events according to the Boolean gates used. The basic event combinations necessary for the higher level event failures are therefore determined. This process is repeated for these higher level events until the top event of the tree is reached, at which point the system failure mode (the top event) is defined by a set of cut sets. The cut sets are then converted into minimal cut sets by removing repeated events and cut sets that are contained within other cut sets.

2.7.1.1 Inclusion-Exclusion Expansion

To determine the probability of a top event represented by a set of cut sets or implicants, it is necessary to perform the inclusion-exclusion expansion.

T is defined as the top event represented by N_{mcs} minimal cut sets, K_i, i=1,…, N_{mcs}, as shown in Equation 2.22.

$$T = K_1 + K_2 + \ldots + K_{N_{mcs}} = \bigcup_{i=1}^{N_{mcs}} K_i \qquad \textbf{2.22}$$

The probability that *T* occurs can then be expressed as the probability that at least one of its minimal cut set occurs, as shown in Equation 2.23.

$$P(T) = P\left(\bigcup_{i=1}^{N_{mcs}} K_i \right) \qquad \textbf{2.23}$$

The inclusion-exclusion expansion must be performed on the minimal cut sets, as defined by Equation 2.24, in order to correctly calculate the top event probability.

$$P(T) = \sum_{i=1}^{N_{mcs}} P(K_i) - \sum_{i=2}^{N_{mcs}} \sum_{j=1}^{i-1} P(K_i \cap K_j) + \sum_{i=3}^{N_{mcs}} \sum_{j=2}^{i-1} \sum_{k=1}^{j-1} P(K_i \cap K_j \cap K_k) - \cdots + (-1)^{N_{mcs}-1} P(K_1 \cap K_2 \cap \ldots \cap K_{N_{mcs}}) \quad \textbf{2.24}$$

The higher order terms of the inclusion-exclusion expansion become negligible when the probability of each individual basic event occurring is very small. In this case the Rare Event Approximation shown in Equation 2.25 may be used. It consists of the first term from Equation 2.24 and gives an upper bound on the actual probability.

$$P(T) \approx \sum_{i=1}^{N_c} P(K_i) \quad \textbf{2.25}$$

A lower bound on the probability is found by using only the first two terms from Equation 2.24 and is known as the Lower Bound Approximation, as shown in Equation 2.26.

$$P(T) \leq \sum_{i=1}^{N_{mcs}} P(K_i) - \sum_{i=2}^{N_{mcs}} \sum_{j=1}^{i-1} P(K_i \cap K_j) \quad \textbf{2.26}$$

2.7.2 BDD Based Quantification Methods

Each path through the BDD from the root (top) node to a terminal node represents a unique mapping of Boolean values to the variables (represented by the nodes) on the path, and the terminal node value is the structure function's output for this mapping. The Boolean value mapped to each variable is indicated by the variables edge the path passes through, with the 1 edge representing the variable's occurrence event and the 0 edge indicating the variable's non-occurrence event. The combination of events existing on a path terminating at a terminal 1 node is an implicant for the structure function.

The probability of the function represented by the BDD evaluating to 1, occurrence of the top event, is therefore found from the sum of the probabilities of these mutually exclusive implicants. Assuming that all node variables are independent, the probability of a node *F*, $P(F)$, is evaluated through Equations 2.27 to 2.29, where *x*, *F0* and *F1* are the node's variable, failure child and success child respectively.

$P(F) = 1$, if F is a terminal 1 node. **2.27**

$P(F) = 0$, if F is a terminal 0 node. **2.28**

$P(F) = P(F1) + \big(1 - P(x)\big)\big(P(F0) - P(F1)\big)$, **2.29**

if F is an *ite* node.

BDDs can be evaluated very efficiently since the probability of each BDD node, which represents part of the systems reliability structure, depends only on the probability of the failure and success events of its variable and the probabilities of its two child nodes. The probability of each node is only calculated once and is then cached to be used in the calculations of all nodes which have that node as a child.

2.7.3 Markov Model Methods

Quantitative analysis cannot be performed directly from the structure function (or its representation) on systems that include repair queuing, standby redundancy and common cause failures. This is because such systems contain basic events whose state depends on the state of other basic events (i.e. they are not independent).

These systems may be analysed using the Markov modelling technique. A Markov model [10] is a form of directed graph consisting of nodes and transitions.

Each node represents a discrete system state. The system states are the finite set of mutually exclusive and exhaustive conditions in which the system can reside, defined in terms of the states of the basic events of the system. The transitions represent the transfer of the system between these states caused by the failure or repair of a basic event. The system's structure function is then used to determine the subset of the system states that represent system failure.

Markov models are only suitable to analyse systems that meet the following requirements:

i. The basic event failure behaviour must not vary with time; the failure and repair rates must be constant, otherwise known as stationary or homogeneous processes. The failure and repair time must therefore be represented by exponential probability distributions.
ii. The future behaviour of the system must not be influenced by its past behaviour; the system must lack memory.
iii. The states of the system must be identifiable, where a system state denotes a vector of the system's component states (whether working or failed).

Requirement *i* may however be relaxed when numerical solution schemes, such as Runge-Kutta [19], that can model time dependant transition rates are used [20]. When a numerical solution scheme is used, transition rates can be globally time dependant, i.e. dependent on the time elapsed from some globally specified time such as the start time at which system operation begins.

An example Markov diagram for a two component system is shown in Figure 2.6, where λ_x is the failure rate of basic event x and υ_x is the repair rate of basic event x.

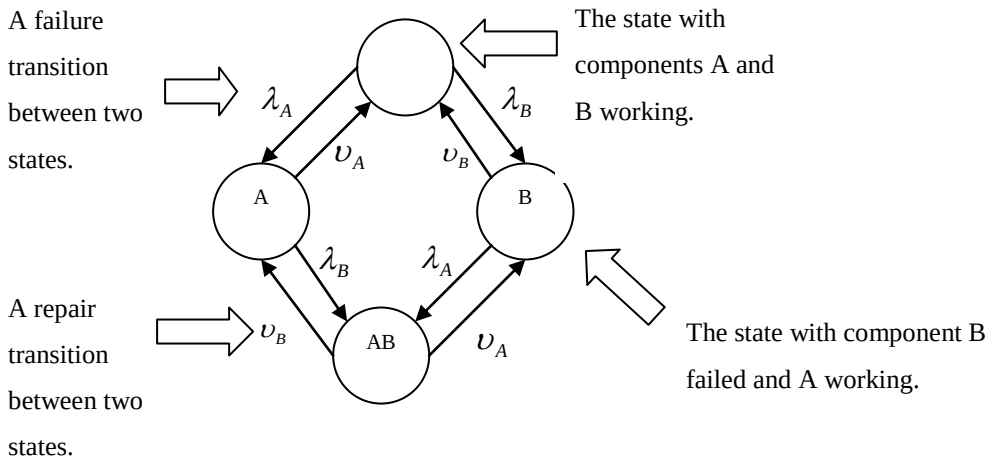


Figure 2.6 - Example Markov Diagram.

One of the major drawbacks of Markov models is that the number of nodes can become huge for systems with large numbers of basic events, making analysis unfeasible due to the computational expense. This is known as the state explosion problem.

2.7.4 Petri Net Models

Another method, especially well suited to modelling complex repairable systems, is Petri Nets. A Petri Net [21] is a directed bipartite graph in which the nodes represent transitions, represented by bars in a graphical representation, and places which are represented by hollow circles. Directed arcs linking places to transitions are known as inputs and those connecting transitions to places are known as outputs. In addition, multiple input or output arcs can link the same place and same transition, with the number of arcs known as the multiplicity, often represented as a single arc with a backslash through it and a positive integer denoting the multiplicity. Places may contain 0 or more tokens, represented by filled circles, and it is the distribution of tokens through the net, known as the net marking, that determines the state of the system. When the number of tokens in a place matches or exceeds the number of input arcs, the transition is enabled, and it may then fire, in which case the tokens are consumed from the input places, and deposited in the output places - thus altering the marking of the net and therefore the state of the system. The firing of the transition can be associated with a time delay, with the length of the delay often described through a distribution. The number of tokens consumed from the input place is equal to the number of input arcs and the number of token deposited in the output place is equal to the number of output arcs. Only one transition can occur at any instant of time, regardless of the number of transitions that are enabled. Figure 2.7 shows a simple Petri net with a transition with three input arcs from two places and a single output arc to another place. In Figure 2.7a there are tokens for each of the input arcs and therefore a

transition occurs, the tokens are consumed and a token deposited at the end of the output arc as shown in Figure 2.7b.

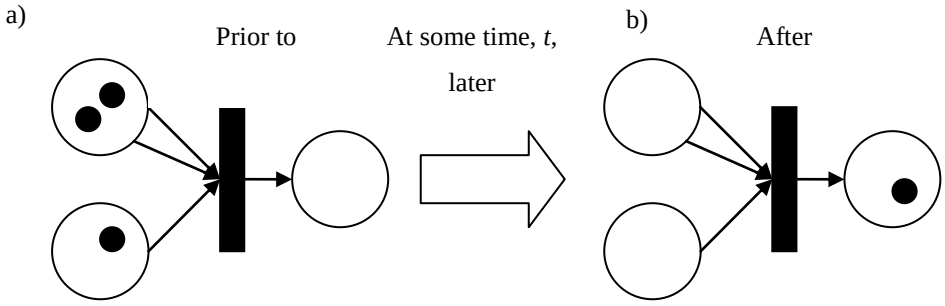


Figure 2.7 - Petri net transition

A fault tree may be converted into a Petri Net and Figure 2.8a and Figure 2.8b show how a Petri net can be used to represent OR and AND gates respectively, where circles represent places, bars represent transitions and arrows represent arcs. In the simplest case a system fault tree is modelled as a Petri net with instantaneous gate transitions and the transition timings of basic events determined through simulation. A token entering the place representing the fault tree's top event then signifies system failure.

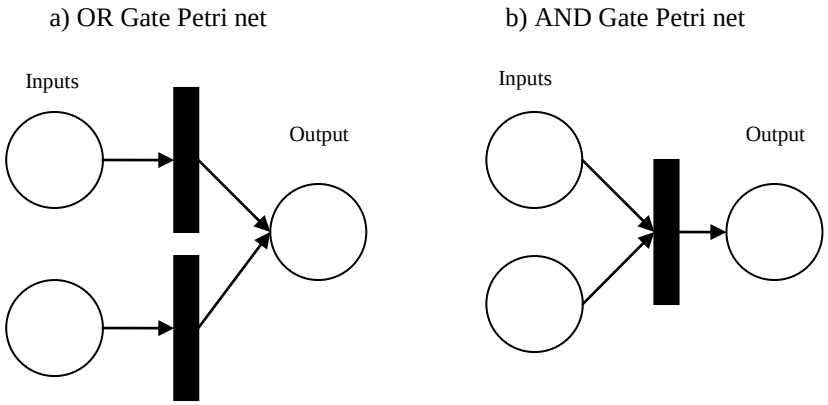


Figure 2.8 - Petri nets of common fault tree gates.

Many other additional features have been added to Petri net models, some that improve the representational power such as coloured tokens [22], and others that improve the expressiveness such as inhibitor arcs that block a transition from occurring.

2.7.5 Monte Carlo Simulation

Monte Carlo simulation [23] is often used to analyse systems that are either too complex or too large for analytical techniques. In this technique, an outcome for the system is determined through randomly generated numbers (typically an algorithm is used to generate pseudo random numbers) that are used to sample from probability distributions in a model of the system reliability. For example a random time, *t*, from an exponential or Weibull distribution can be obtained from Equation 2.30 and Equation 2.31 respectively, where *X* is a uniformly distributed random variable between 0 and 1 [13]. Each simulated outcome is known as a trial, a large number of which are recorded over a typical simulation. The distribution of simulated outcomes should then follow the outcome distribution of the simulated system, provided that a sufficient number of trials have been computed. Reliability metrics are then computed from the generated outcome data.

$$t = -\lambda \ln(X) \qquad \textbf{2.30}$$

$$t = \eta(-\ln(X))^{\frac{1}{\beta}} \qquad \textbf{2.31}$$

Simulation used to model component reliability together when modelling system reliability through fault trees or Petri Nets. The main benefits of simulation over analytical solution schemes are that it is easy to apply, can be used to model very complex scenarios and has the ability to generate very detailed statistics. The disadvantages are that results are not exact and that generating a sufficient number of simulated outcomes has a high computational cost. Its independence from analytical methods makes it ideal for verifying their accuracy through a comparison of results.

2.8 Phased Mission Analysis

Many real world systems operate in what are known as phased missions [24]. A system operates in a phased mission if the reliability structure changes at the transition points between consecutive time periods, known as phases, of the system's operation. Common causes of the differences before and after a transition time are changes in the system configuration or mode of operation. Each phase has a certain duration that is normally fixed but may also be modelled by a random variable. At the end of the duration of a phase, a phase transition occurs, the current phase ends and the system reliability structure is replaced by that of the next phase. It is usually assumed that phase transitions occur instantaneously such that no component failures can occur whilst it takes place. The type of phased mission studied in this document are defined as successful if all phases are completed without system failure and failed if system failure occurs in any phase, at which point the mission terminates immediately. For details on phased missions with combinatorial phase requirement see Misra [25].

An example of a phased mission is a landing gear system used during a commercial airliner flight where the phases include take-off, climb, cruise, descent and landing; each of these phases have different component failure combinations resulting in phase failure due to the different configuration and role of the system. A chart showing the altitude and reliability of a landing gear system during a phased mission is shown in Figure 2.9.

A system operating in a phased mission can fail in one of two ways. It can fail within a phase if failures of its components cause a system failure combination to occur and this is known as an in-phase failure. Alternatively, the system may be functioning at the end of a certain phase, and without additional component failures, fail on transition to the next phase due to the change in the system reliability structure resulting in a system failure combination, present in the previous phase, to cause system failure under the new phase conditions. This second failure type is known as a transitional failure. Phase failure alone, when in-phase or transitional failure is not specified, includes failures on transition to and during the phase.

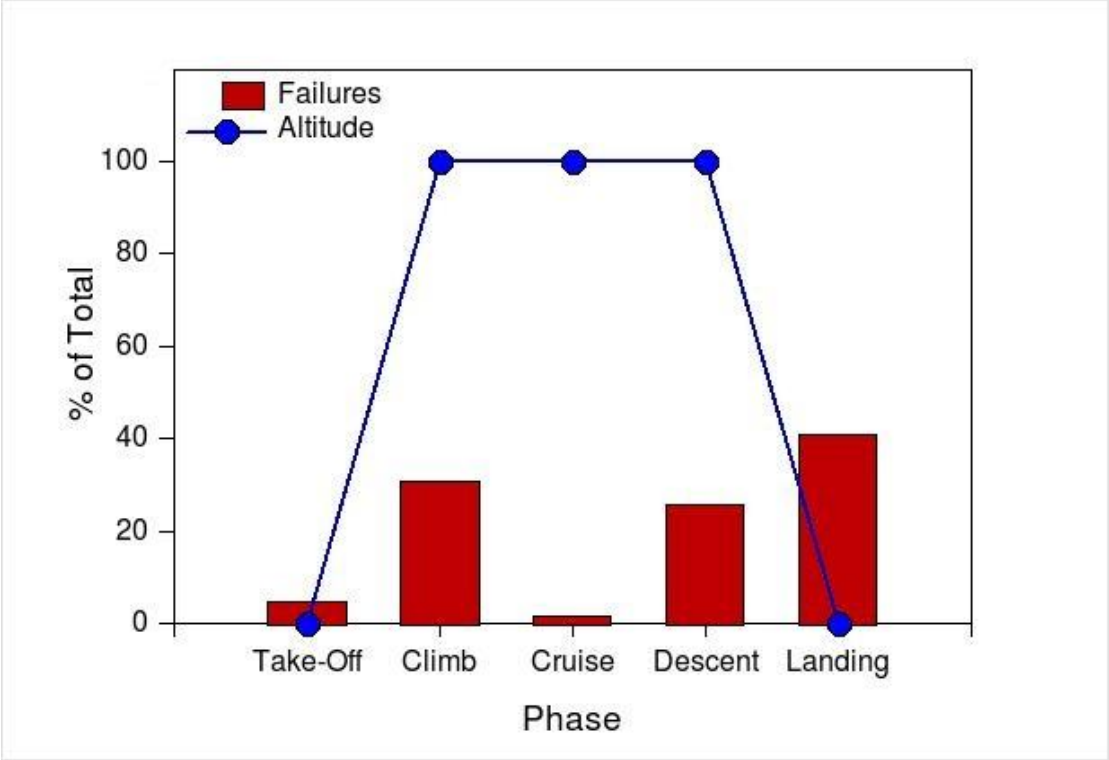


Figure 2.9 – Landing gear system altitude and reliability in a phased mission

2.8.1 Phased Mission Fault Trees

Phased mission fault trees are used to represent the failure or success of a phased mission over a certain period. The most commonly used fault trees are shown in Figure 2.10, where *N* is the number of phases in the mission. The complement of a phase fault tree, formed by placing a NOT gate as a parent to the existing top gate, is known as the phase success tree or phase dual tree. This is equivalent to replacing AND gates with OR gates, OR gates with AND gates and component failure basic events with component success basic events.

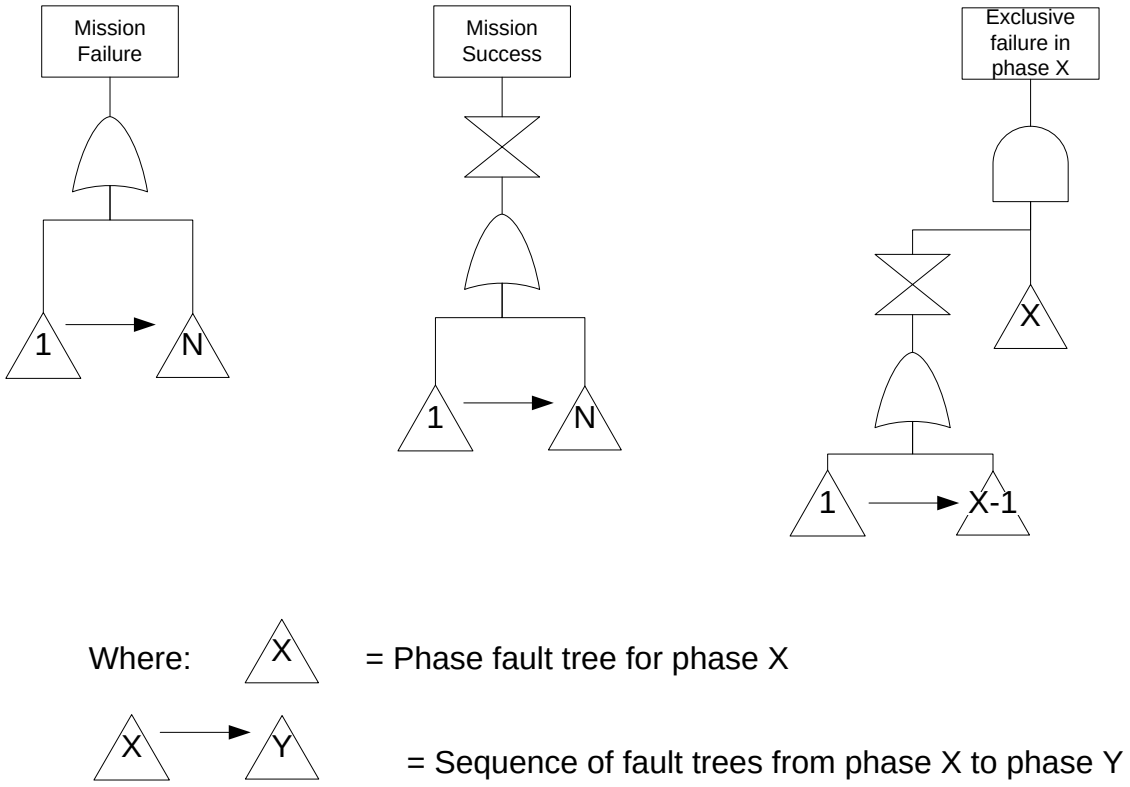


Figure 2.10 – Common mission fault trees

2.8.2 Calculation of system failure during any period of the mission

The probability of system failure between times T_1 and T_2 in a non-repairable phased mission, denoted as Q^{T_1,T_2}, can be calculated from Equation 2.32, where x_i represents the failure event for the system.

$Q^{T_1,T_2} = R(x_i, T_2) - R(x_i, T_1)$, where $T_2 \geq T_1$ **2.32**

The values of T_1 and T_2 for the calculation of system failure during phase *j*, Q_j , phase *j* in-phase system failure, Q_j^P, system failure on transition to phase *j*, Q_j^T, and failure during a mission of *N* phases, Q_{Miss}, are given in Table 2.2, where t_j is the time at which phase *j* ends, t_j^- is the time just before the phase change at the end of phase *j* (i.e. $t_j - dt$) and t_j^+ is the time just after (i.e. $t_j + dt$).

Table 2.2 - The T_1 and T_2 times for calculating probabilities of system failure in various time periods of a phased mission.

Period of phased mission in which system failure occurs	Symbol	Equation 2.32 T_1 value	Equation 2.32 T_2 value
Overall phase j	Q_j	t_{j-1}^-	t_j^-
In-Phase phase j	Q_j^P	t_{j-1}^+	t_j^-
Transition to phase j	Q_j^T	t_{j-1}^-	t_{j-1}^+
Mission of *N* phases	Q_{Miss}	t_0	t_N

Note that whilst system failures on transition to phase j occur at time t_j , the probability is measured through Equation 2.32 as the probability of system failure occurring in an infimintesimal time period that contains t_j .

2.9 Summary

Risk and reliability engineering is an important field that advances methods for measuring and improving the risk and reliability of systems. Risk is the product of the probability of occurrence of some undesired event and its consequences. The various reliability metrics are useful for presenting reliability performance and comparing the different aspects of reliability between systems. Many methods are available that can be used to represent a system's reliability structure and the choice depends on both the system being analysed and the desired type of analysis. Different quantitative analysis methods, ranging from those that give exact results, such as binary decision diagrams, to the more flexible Monte Carlo simulation can be used, each with its own advantages and disadvantages. Phased missions, where the system's reliability structure varies over consecutive time periods, are a common occurrence in the real world and research into their reliability is an important topic.

3 Literature Review of reliability methods for non-repairable phased missions

3.1 Introduction

This chapter covers existing research on the reliability analysis of non-repairable systems undergoing phased missions with independent component failures. Systems in many real world phased missions are non-repairable, e.g. it is not possible to repair landing gear components during a passenger aircraft flight. The research covered begins with the earliest methods that are based on transformations and utilise methods developed for non-phased missions. It then covers methods based on the use of Boolean phase algebra and finishes with the most recent methods that utilise binary decision diagrams.

3.2 Early Phased Mission Methods

The earliest method for analysing the reliability of non-repairable phased mission was developed by Esary and Ziehms [3]. They present a method that transforms a multi-phase mission into an equivalent single phased mission for the purpose of calculating a system's mission reliability exactly. The data required to calculate the mission reliability using this method are the phase fault trees, which are assumed to be coherent, and the probability of failure in each phase for each component conditional on the component surviving until that phase. These conditional phase failure probabilities are easily determined from the components failure probability density or cumulative failure distribution functions.

Esary and Ziehms show that attempting to calculate mission reliability by forming fault trees for each phase, calculating the reliability of each phase individually using standard fault tree methods and then multiplying the individual phase reliabilities together does not give the correct result. This is due to the invalid implicit assumption that failure probabilities of components in phases are statistically independent of the system performance in earlier phases, which would only be true if none of the phases contained common components. The resulting inaccuracy from this assumption can be significant.

Esary and Ziehms method (EZ method) transforms the multi-phase mission into an equivalent single phased mission. In order to achieve this they note that the added complexity inherent in phased missions is because a component's performance in each phase depends on its performance in previous phases. Specifically, its performance in a given phase depends on whether it has functioned successfully, without failure, in all previous phases. This is based on the fact that the components are non-repairable and therefore to be in a working state in a given phase, they must have worked continuously in all previous phases. Hence where the system reliability structure is represented by a reliability block diagram, a component failure event in a given phase can be replaced by a series arrangement of events for each phase up to and including the current phase, where each event represents the failure of the component in that specific phase, i.e. the event that it fails within that phase conditional on it surviving until that phase. Where the system reliability structure for a phase *i* is represented by a fault tree, a basic event in that tree is replaced by a logical OR gate with input basic events representing component failure in each individual phase up to and including phase *i*. For example, Figure 3.1a shows a fault

tree for system failure in phase 2 prior to transformation and Figure 3.1b shows the same fault tree afterwards, where $x_i(j,k)$ represents the occurrence of event x_i between times t_j and t_k (i.e. during phase *k* if *k=j+1*). For example, in the fault tree shown in Figure 3.1a, basic event *A* is replaced by an OR gate with inputs of A(0,1) and A(1,2) in the transformed fault tree shown in Figure 3.1b.

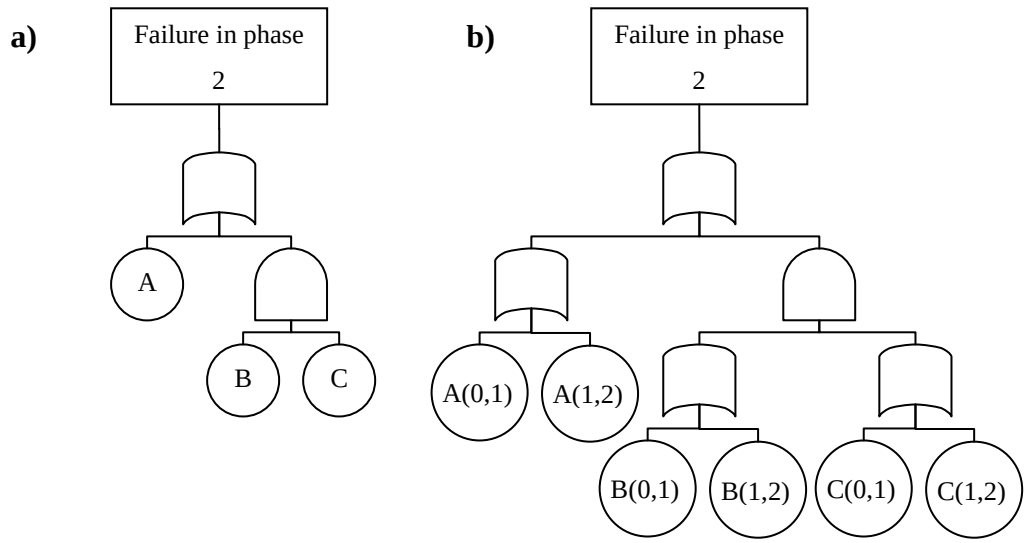


Figure 3.1 - Example of a phase fault tree transformation with the EZ method

The mission unreliability is then represented through an OR gate with inputs of the individual phase fault trees, since failure in any phase results in failure of the mission. This results in a new, single phase, system that has the same mission reliability as the original system. The number of basic events in the resultant system, known as the equivalent system, will be increased, at most, by a factor equal to the number of phases in the mission. This can lead to a huge increase in the number of cut sets for mission failure, severely limiting the size of the systems that may be analysed due to the computational burden imposed.

The minimal cut set cancellation technique can be applied prior to transforming the system and results in a reduced number of cut sets and overall computational effort. A minimal cut set can be cancelled, i.e. omitted from the list of minimal cut sets for that phase, if it contains a minimal cut set of a later phase. This relies on the fact that, for a non-repairable system, occurrence of the cancelled cut set means that the cut set it contains will also occur, causing system failure in the later phase anyway. The mission unreliability is therefore left unchanged by the application of the technique. Applying the cut set cancellation technique reduces the number of cut sets that must be computed by the fault tree algorithm and results in faster analysis.

Burdick, Fussell, Rasmuson and Wilson [26] reviewed some approximation methods that allow the transformation into an equivalent single phase system to be avoided, hence reducing the computation burden. The methods rely on obtaining the reliability of each phase separately and using these to estimate the overall mission reliability. The methods are all conservative, giving an upper bound on the exact unreliability value, and can therefore be used to determine if the systems reliability performance is above a certain level. However, the use of conservative approximations can lead to the over design of a system, so whilst useful as fast estimates they should be used with care.

The INEX approximation involves carrying out an inclusion-exclusion expansion of the minimal cut sets of each phase, using unconditional basic event unreliabilities, to obtain the approximate unreliability of each phase. The appoximate phase *j* system unreliability found through this method is denoted as Q_{j-INEX}. The overall INEX mission unreliability approximation, $Q_{Miss-INEX}$, is then found by deducting the product of the individual phase reliabilities (1 – unreliability) from one. An alternative that gives an upper bound on $Q_{Miss-INEX}$, is to sum the INEX unreliability approximations from each phase as shown in Equation 3.1, where *N* is the total number of phases in the mission.

$$Q_{Miss-INEX} \leq \sum_{j=1}^{N} Q_{j-INEX} \qquad \textbf{3.1}$$

The INEX approximation can also be used after the cut set cancellation method has been applied, resulting in an equal or lower reliability approximation than that obtained without cancellation. This is known as the INEX-CC approximation, $Q_{Miss-INEX-CC}$.

The second conservative method uses the minimal cut set bound and is known as the MCB approximation. The probability of cut set *k* occurring in phase *j* is calculated by the multiplication of its basic events, using unconditional failure probabilities, through Equation 3.2, where $r_{k,j}$ is the probability of the occurrence of cut set *k* from the phase *j* fault tree, x_i is basic event *i* of $k_{k,j}$ basic events in cut set *k* from the phase *j* fault tree and $P(x_i(0,j))$ is the unconditional probability of x_i occurring at or before the end of phase *j*.

$$r_{k,j} = \prod_{i=1}^{k_{k,j}} P(x_i(0,j)) \qquad \textbf{3.2}$$

The MCB approximation of the reliability in phase *j* of the phased mission, R_{j-MCB}, is then calculated as the probability of no phase cut sets occurring, using Equation 3.3, where N_{MCS_j} is the number of phase *j* cut sets.

$$R_{j-MCB} = \prod_{k=1}^{N_{MCS_j}} (1 - r_{k,j}) \qquad \textbf{3.3}$$

The MCB approximation of the mission unreliability, $Q_{Miss-MCB}$, is then calculated as one minus the product of the individual phase MCB reliability approximations as shown in Equation 3.4.

$$Q_{Miss-MCB} = 1 - \prod_{j=1}^{N} R_{j-MCB} \qquad \textbf{3.4}$$

Again, prior application of the minimal cut set cancellation rules give a further approximation, known as the MCB-CC approximation, $Q_{Miss-MCB-CC}$.

Zhiems [27] showed that the ordering of the approximations is given by Equation 3.5, where Q_{Miss} is the exact mission unreliability calculated through the EZ transformation method.

$$Q_{Miss} \leq Q_{Miss-INEX-CC} \leq \begin{matrix} Q_{Miss-MCB-CC} \\ Q_{Miss-INEX} \end{matrix} \leq Q_{Miss-MCB} \qquad \textbf{3.5}$$

3.2.1 Conclusions

Esary and Zhiems presented a method capable of determining the reliability of a mission from the phase fault trees and conditional failure probabilities of the basic events in each phase of the mission.

The major disadvantage the method has is that due to the increase in the size of the system fault tree caused by the splitting of the component unreliability in a given phase into a series system of conditional probabilities, the number of cut sets can increase significantly, resulting in large computational requirements. A number of estimation methods were presented by Burdick et al that give upper bounds on the mission unreliability at lower computational expense than the exact value from the EZ method.

A limitation of these early methods are that they can only calculate the overall mission unreliability and not the individual phase unreliabilities.

3.3 Boolean phase algebra based methods

Instead of replacing a component event in a particular phase with events representing its performance in each phase, as used in the EZ method with resultant increase in the number of basic events and computational expense, an array of methods have been developed that instead use Boolean phase algebra to resolve the phase dependencies between events that occur when analysing phased missions. The Boolean phase algebras in these methods resolve combinations of dependant events into minimal combinations of independent events so that the probability of a combination can be found from the product of the individual event probabilities; although some of the phase algebras leave certain dependencies unresolved. These algebra rules are used in conjunction with the standard Boolean algebra rules given in Equations 2.18 to 2.20. They offer a significant performance advantage over the earlier methods and some can also be used to directly obtain the probability of failure in an individual phase.

The methods discussed in this section have been split into two sets; those whose unreliability evaluations are based on the inclusion-exclusion expansion based methods and the more recent methods that instead use the binary decision diagram technique to give improved performance.

3.3.1 Inclusion-Exclusion expansion based methods

Dazhi and Xiaozhong [1] and Kohda, Wada and Inoue [28] developed methods that applied phased mission analysis methods to determine the probability of accident sequences. Not only did they advance the work of [3] by removing the significant increase in the number of cut sets and resultant computational expense caused by removing the need to model the same component failure in different phases as different basic events, but by developing methods that could analyse accident sequences containing the success of subsystems, they developed methods that also solved the problem of directly determining the probability of failure within a particular phase although this was not the intention of their work.

Determining the probability of failure within an individual phase is very useful for many reasons. For example, there are often different consequences of failure depending on the phase and thus quantifying the risk of a mission requires knowledge of the individual phase failure probabilities. An example of this is the failure of an aircraft landing gear system during landing is likely to have greater consequences than its failure during taxi on the runway. Whilst possible to calculate using the EZ method, it requires the calculation of the difference between the mission unreliability up to the start and up to the end of the phase and therefore involves two phased mission unreliability evaluations.

As stated earlier, the methods in [1] and [28] were developed for the analysis of accident sequences. An accident sequence consists of an initiating event, the failure of one or more subsystems and possibly the success of other subsystems. Each sequence is defined by the combinations of the subsystems that fail and those that operate successfully. Usually an event tree used to represent the possible accident sequences whilst fault trees are used to represent the combinations of basic events that cause failure of each subsystem. Failure of a system in a certain phase of a phased mission, known as exclusive phase failure, requires its successful operation in preceding phases, so that the phase is reached, and then failure occurring at some point in that phase. It is therefore clear that the methods described in [1] and [28] can be applied to obtaining the probability that failure occurs in a certain phase of a phased mission. Essentially, the problem of determining the probability of failure in a certain phase can be treated as an accident sequence with an initiating event probability of 1 and a sequence consisting of the successful operation of a set of subsystems (the prior phases) followed by the failure of a subsystem (the phase being analysed). The subsystem fault trees are therefore replaced by the phase fault trees, with earlier phases successful and failure occurring in the final phase.

Both introduced new Boolean phase algebra to resolve the statistical dependences between failures of a component in different phases. Due to this, the application of their methods requires the use of specific fault tree codes that implement the Boolean algebra rules.

The Boolean algebra rules introduced by [1], which they name the generalized intersection and union concept, consider the time of component failures and allow faster evaluation of the mission unreliability than the EZ method as well as the calculation of the unreliability in individual phases. Their method will be described using phased mission rather than accident sequence terminology (e.g. phase instead of subsystem) and with slight modifications (e.g. initiating events, not relevant to phased mission analysis, have been omitted). In addition to

the usual phased mission assumptions and those stated in the introduction, this method assumes that phase fault trees are coherent.

Instead of considering the same component failure in different phases as different basic events, as is the case with the EZ method, in this method a component failure in a phase is represented by the basic event that the component fails up to the end of that phase. For example, the failure event for component B appearing in the phase 3 fault tree is represented by the basic event B(0,3), since that event will occur if it has failed at any time up to the end of that phase. The phase dependencies between basic events representing component failures in different phases are then resolved through the application of phase Boolean algebra rules. Since the assumption is made that phase fault trees are coherent, the Boolean algebra rules only cover Boolean OR and Boolean AND operations between failure basic events, as success basic events will not be encountered. The Boolean algebra OR reduction rule is given by Equation 3.6 and the Boolean algebra AND reduction rule is given by Equation 3.7.

$A(0,k) \vee A(0,j) = A(0,j)$, where $j \geq k \geq 1$ **3.6**

$A(0,k) \wedge A(0,j) = A(0,k)$, where $j \geq k \geq 1$ **3.7**

The two rules are based on the fact that $A(0,j) = A(0,1) \vee A(1,2) \vee \ldots \vee A(j-1,j)$ - the transformation applied with the EZ method. Therefore, when split into individual phase basic events and $j \geq k \geq 1$, $A(0,j)$ contains all the basic events contained in $A(0,k)$, hence the reduction shown in Equation 3.6 applies to a logical OR between the basic events, and the basic events common to both $A(0,k)$ and $A(0,j)$ are just $A(0,k)$, thus Equation 3.7 applies to a logical AND between them.

The failure conditions for phase *j*, X_j, can be expressed as the union of the N_{mcs_j} phase *j* minimal cut sets since failure occurs when at least one cut set occurs. This is shown by Equation 3.8, where $C_{k,j}$ is the *k*th cut set for phase *j*.

$$X_j = \bigcup_{k=1}^{N_{mcs_j}} C_{k,j} \quad \textbf{3.8}$$

X_j can also be expressed in terms of the individual component failure events from each cut set, as shown in Equation 3.9, where *l* is a component failure event from the *k*th phase *j* cut set.

$$X_j = \bigcup_{k=1}^{N_{mcs_j}} \left(\bigcap_{l \in C_{k,j}} l \right) \quad \textbf{3.9}$$

A mission fails if system failure occurs in any phase, therefore Equation 3.10 expresses the mission unreliability as the probability of the union of failure in any of the mission phases.

$$Q_{Miss} = P\left(\bigcup_{j=1}^{N} X_j\right) \qquad \textbf{3.10}$$

To calculate the probability of exclusive failure in phase *j*, the first step is to set the top events of the fault trees from the *j-1* earlier phases, which represent the success phases, as inputs to a logical OR gate and determine the minimal cut sets from this fault tree. This fault tree represents mission failure in any of the success phases, i.e. the mission ending before phase *j* is reached. The second step is to find the minimal cut sets from the phase *j* fault tree. The probability of failure in phase *j*, Q_j, is then the probability of the occurrence of any of the N_{MCS_j} minimal cut sets from phase *j* minus the probability that any of the N_{MCS_j} minimal cut sets from phase *j* occur along with any of the minimal cut sets from the earlier success phases, as shown in Equation 3.11.

$$Q_j = P\left(\bigcup_{k=1}^{N_{MCS_j}} C_{k,j}\right) - P\left(\left(\bigcup_{k=1}^{N_{MCS_j}} C_{k,j}\right) \cap \left(\bigcup_{l=1}^{j-1} \bigcup_{k=1}^{N_{MCS_l}} C_{k,l}\right)\right) \qquad \textbf{3.11}$$

The Boolean algebra rules will automatically lead to the cancellation of cut sets that contain a minimal cut set of a later phase in Equations 3.8 to 3.11.

Dazhi and Xiaozhong note that Equation 3.11 is computationally expensive due to the large number of minimal cut sets and because the inclusion-exclusion expansion must be used to evaluate the exact probabilities of the sets of cut sets. They therefore present a method that approximates the exclusive phase *j* failure probability based on the rare event assumption and the concept of superset minimal cut sets. Minimal cut set *A* is defined as a superset of minimal cut set *B*, if each element in *A* is also in *B*. It is named a superset (even though it is a subset in terms of its elements) since all component failure combinations that cause *B* to occur also cause *A* to occur. It gives an approximation that is greater than or equal to the actual unreliability. This approximation is calculated through Equation 3.12, where $C_{k,j}$ for $k = 1, \dots, k_1$ are the phase *j* cut sets that have no supersets in the success phase cut sets, $C_{k,j}$ for $k = k_1 + 1, \dots, N_{MCS_j}$ are the phase *j* cut sets with at least one superset in the success phase cut sets and $SUP_{k,j}$ is the set of success phase cut sets that are supersets of the *k*th failure phase cut set, $C_{k,j}$. In Equation 3.12, the probability that the failure conditions for both phase *j* and an earlier success phase occur is calculated as the sum of the probabilities of the intersection of each minimal cut set from phase *j* with its superset minimal cut sets from the success phases. The probability that the phase *j* failure conditions occur is calculated as the sum of the phase *j* cut set probabilities. The probability that exclusive phase failure

occurs in phase *j* is then calculated as the probability that the failure conditions for phase *j* are met minus the probability that both the failure conditions for phase *j* and the failure conditions for an earlier success phase are met, as before. Dazhi and Xiaozhong justify the removal of the terms not present in the unreliability calculation when Equation 3.12 is used compared to when the exact calculation is used, Equation 3.11, by showing that they are negligible, assuming rare events, through a series of examples. Since the approximation is based on the assumption of rare events, it can be inaccurate when cut sets with a high probability of occurrence exist, in which case Equation 3.11 should be used instead.

$$Q_j \approx \sum_{k=1}^{k_1} P\left(C_{k,j}\right) + \sum_{k=k_1+1}^{N_{MCS_j}} \left(P\left(C_{k,j}\right) - P\left(SUP_{k,j} \cap C_{k,j}\right)\right) \qquad \textbf{3.12}$$

Kohda, Wada and Inoue [28] presented a new method for application to accident sequences that can also be used to obtain exact phase failure probabilities for a phased mission. It expresses the time requirements for a basic event explicitly and reduces the computational burden present in the earlier methods. Again, their method is described here in terms of phased missions instead of the accident sequence terminology used in their paper and is based upon the usual assumptions for non-repairable phased missions, therefore, unlike Dazhi and Xiaozhong's methods discussed earlier, it can be used with non-coherent phase fault trees.

The method requires the derivation of the path sets for each of the *j-1* preceding success phases, in addition to the failure implicants (rather than cut sets, since non-coherent fault trees are allowed) for phase *j*, in order to obtain the exclusive phase *j* failure probability. Whilst a phase failure implicant is a combination of basic events that represent system failure conditions for that phase, a phase path set is a combination of basic events that result in system success for that phase. Phase path sets are found from the phase dual tree [29].

Kohda, Wada and Inoue introduced a phase Boolean algebra that operates directly on the start and end times of the basic event time periods. Since the event that a component works successfully up to a certain time is equivalent to the component failing at some point after that time, they represent basic event that a component, *A*, works successfully until time t_j, as $A(j,\infty)$. $A(j,\infty)$ is therefore equivalent to $\bar{A}(0,j)$ in the usual notation. This transformation allows both failure and success basic events to be represented through the start and end times of the time period of a failure event, reducing the number of Boolean algebra rules that must be specified. The Boolean algebra rules are shown in Equations 3.13 to 3.15.

$$A(i,j) = 0, \qquad if\ t_j \le t_i \qquad \textbf{3.13}$$

$$A(i_1,j_1) \vee A(i_2,j_2) = A\left(min(i_1,i_2), max(j_1,j_2)\right), \qquad if\ t_{i_1} \le t_{j_2}\ or\ t_{i_2} \le t_{j_1} \qquad \textbf{3.14}$$

$$A(i_1, j_1) \wedge A(i_2, j_2) = A\big(max(i_1, i_2), min(j_1, j_2)\big) \quad \textbf{3.15}$$

Equation 3.13 expresses the fact that the event never occurs, since the time period for the event has no duration, therefore any combination containing such an event can be removed from the analysis as it has zero probability. Equations 3.14 and 3.15 implement the Boolean OR and AND rules respectively and extend the equivalent rules given in Equations 3.6 and 3.7 by Dazhi and Xiaozhong, by also resolving combinations that include basic events representing component success.

The success conditions for phase *j*, S_j, can be expressed as the union of the phase *j* minimal path sets since success occurs when at least one path set occurs, as shown by Equation 3.16, where $P_{k,j}$ is the *k*th minimal path set from phase *j* and N_{MPS_j} is the total number of minimal path sets for phase *j*.

$$S_j = \bigcup_{k=1}^{N_{MPS_j}} P_{k,j} \quad \textbf{3.16}$$

S_j can also be expressed in terms of the individual component success events as shown in Equation 3.17, where *l* is a component success event from the *k*th phase *j* path set.

$$S_j = \bigcup_{k=1}^{N_{MPS_j}} \bigcap_{l \in P_{k,j}} l \quad \textbf{3.17}$$

The mission unreliability is calculated through Equation 3.10, as in [1]. The complement of the mission unreliability, the mission success probability or reliability, R_{Miss}, can be calculated directly through Equation 3.18.

$$R_{Miss} = P\left(\bigcap_{j=1}^{N} S_j\right) \quad \textbf{3.18}$$

The probability that exclusive failure in phase *j* occurs is now calculated through Equation 3.19. Comparing Equation 3.11 with Equation 3.19 shows the two different approaches for calculating the exclusive phase failure probability used by Dazhi and Xiaozhong and Kohda et al. In the method from Dazhi and Xiaozhong, the probability of exclusive phase failure in phase *j* is calculated by deducting the probability that the failure conditions for both phase *j* and at least one earlier phase are met from the probability that the failure conditions

for phase *j* are met. In the method from Kohda et al, a direct representation of the success of the *j-1* earlier phases and failure in phase *j* is used in the probability calculation.

$$Q_j = P\left(\left(\bigcap_{i=1}^{j-1} S_i\right) \cap X_j\right) \qquad \textbf{3.19}$$

Kohda et al demonstrate the improved performance of their method over the EZ method in a series of examples that show calculations of both the mission unreliability and probability of failure in a specific phase of the mission.

Another method that utilises a phase Boolean algebra to calculate the mission unreliability, but unlike the others is unable to calculate exclusive phase failure probability directly, was presented by Somani and Trivedi [30]. Like the other methods using Boolean phase algebra, it avoids the transformation of the mission into a single phase and resultant increase in the number of basic events and computational expense.

Somani and Trivedi describe the possible cases that can occur on transition between phases due to the different reliability structures between phases:

1. A combination of component failures does not lead to system failure in both phases *i* and *i* + 1.
2. A combination of component failures leads to system failure in both phases *i* and *i* + 1.
3. A combination of component failures does not imply system failure in phase *i* but leads to system failure in phases *i* + 1.
4. A combination of component failures implies system failure in phase *i* but does not imply system failure in phase *i* + 1.

The third case results in failure on transition between phases and can be considered as a failure combination for both phases, i.e. this can be considered in the same way as case 2. The first three cases can be treated in the same way as the situation where the reliability structure of the system remains the same between phases. The effect on mission reliability of these cases can be determined by evaluating the fault tree of the last phase of the mission. For the fourth case, where a combination implies system failure in the phase *i* but not in subsequent phases, only the occurrence of the combination until the end of phase *i* needs to be accounted for. The occurrence of such combinations in later phases does not affect the system reliability.

The mission reliability can therefore be split into two parts:

- The common failure combinations (first three cases).
- Phase failure combinations (fourth case).

The contribution to mission unreliability of the common failure combinations is determined by calculating the component availability at the end of the last phase and then evaluating the fault tree for the last phase.

Calculating the contribution to mission unreliability of the phase failure combinations is more complicated. The combinations that fail phase *j* but not subsequent phases, PFC_j, can be found through the intersection of the phase failure implicants and subsequent phase path sets as shown by Equation 3.20.

$$PFC_j = X_j \cap \left(S_{j+1} \cap \ldots \cap S_n\right) \quad \textbf{3.20}$$

Equation 3.20 leads to expressions involving combinations of component failure and success events. Somani and Trivedi present the set of Boolean phase algebra rules shown in Equations 3.21 to 3.26 to resolve the dependencies between these events.

$$\overline{A}(0,i) \wedge \overline{A}(0,j) = \overline{A}(0,j), \text{ where } i < j. \quad \textbf{3.21}$$

$$A(0,i) \wedge A(0,j) = A(0,i), \text{ where } i < j. \quad \textbf{3.22}$$

$$A(0,i) \wedge \overline{A}(0,j) = 0, \text{ where } i < j. \quad \textbf{3.23}$$

$$A(0,i) \vee A(0,j) = A(0,j), \text{ where } i < j. \quad \textbf{3.24}$$

$$\overline{A}(0,i) \vee \overline{A}(0,j) = \overline{A}(0,i), \text{ where } i < j. \quad \textbf{3.25}$$

$$\overline{A}(0,i) \vee A(0,j) = 1, \text{ where } i < j. \quad \textbf{3.26}$$

Note that there is no rule specified in the Boolean phase algebra to resolve a combination of component success up to a certain phase and component failure before some later phase, i.e. $\overline{A}(0,i) \wedge A(0,j)$ where $i < j$. The probability of combinations such as this are calculated through Equation 3.27. The algebra is therefore less complete than that specified by Kohda et al, which resolves all dependencies.

$$P\left(\overline{A}(0,i) \wedge A(0,j)\right) = P\left(A(0,j)\right) - P\left(A(0,i)\right), \text{ where } i < j. \quad \textbf{3.27}$$

The method is also limited to evaluating the mission unreliability or unreliability at the end of a certain phase. This is due to the way in which the method accounts for failure combinations that satisfy the failure conditions for more than one phase, whilst satisfying the success conditions for later phases in the mission. The method accounts for these combinations as phase failure combinations in the latest phase in which they cause failure. The effect of the occurrence of these failure combinations, on the mission unreliability, will therefore be correctly accounted for. However, since their contribution is accounted for only in the final phase for which they are a system failure combination, the method cannot be used to calculate individual phase reliabilities directly.

The overall mission unreliability, Q_{Miss}, is calculated as the probability of the occurrence of a phase failure combination or a common failure combination, as shown by Equation 3.28. To calculate the probabilities of the

phase failure combinations, PFC_j, and the common failure combinations, X_N, they must first be expanded through the inclusion-exclusion expansion.

$$Q_{Miss} = P(X_N) + \sum_{j=1}^{N-1} P\left(PFC_j\right) \quad \textbf{3.28}$$

3.3.1.1 Example

The methods from Dazhi and Xiaozhong, and Kohda et al are demonstrated in this section through the calculation of the probability of exclusive phase failure in the last phase of a simple two phased example mission whose fault trees are shown in Figure 3.2. The method from Somani and Trivedi is not included in this example as it cannot be used to derive exclusive phase failure probabilities directly.

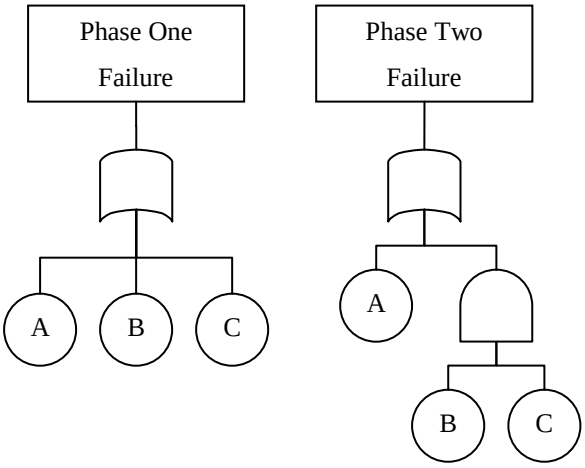


Figure 3.2 - Phase fault trees

The phase one fault tree has the three minimal cut sets shown in Equations 3.29 to 3.31, whilst the phase two fault tree has the two minimal cut sets shown in Equations 3.32 and 3.33.

$$C_{1,1} = A(0,1) \quad \textbf{3.29}$$

$$C_{2,1} = B(0,1) \quad \textbf{3.30}$$

$$C_{3,1} = C(0,1) \quad \textbf{3.31}$$

$$C_{1,2} = A(0,2) \quad \textbf{3.32}$$

$$C_{2,2} = B(0,2) \wedge C(0,2) \quad \textbf{3.33}$$

The components are all assumed to have constant failure rates with the values given in Table 3.1.

Table 3.1 - Component failure rates.

Component	**Failure Rate (failures per hour)**
A	3×10^{-4}
B	5×10^{-4}
C	1×10^{-3}

The durations of phases one and two are 10hrs and 20hrs respectively. The probabilities of component failure in phases 1 and 2 are given in Table 3.2.

Table 3.2 – Component phase failure probabilities.

Component Failure Event	**Probability**
$A(0,1)$	2.9955×10^{-3}
$A(1,2)$	5.9641×10^{-3}
$B(0,1)$	4.9875×10^{-3}
$B(1,2)$	9.9005×10^{-3}
$C(0,1)$	9.9502×10^{-3}
$C(1,2)$	1.9604×10^{-2}

Dazhi and Xiaozhong method

The exclusive phase two failure probability will be first calculated through the exact method given by Dazhi and Xiaozhong, Equation 3.11. The first term in Equation 3.11 requires the union of the minimal cut sets from the failure phase, phase 2, to be determined, as shown in Equation 3.34.

$$\bigcup_{k=1}^{2} C_{k,2} = A(0,2) \vee B(0,2) \wedge C(0,2) \qquad \textbf{3.34}$$

The second term in Equation 3.11 requires the intersection of the unions of the minimal cut sets from the success and failure phases, phases 1 and 2 respectively, to be determined. The union of the success phase minimal cut sets is shown in Equation 3.35.

$$\bigcup_{k=1}^{3} C_{k,1} = A(0,1) \vee B(0,1) \vee C(0,1) \qquad \textbf{3.35}$$

The intersection of the unions of the minimal cut sets from the two phases, Equations 3.34 and 3.35, is shown in Equation 3.36.

$$\bigcup_{k=1}^{2} C_{k,1} \cap \bigcup_{k=1}^{3} C_{k,2} = \big(A(0,1) \vee B(0,1) \vee C(0,1)\big) \wedge \big(A(0,2) \vee B(0,2) \wedge C(0,2)\big) \qquad \textbf{3.36}$$

$$= A(0,1) \vee \big(A(0,1) \wedge B(0,2) \wedge C(0,2)\big) \vee \big(A(0,2) \wedge B(0,1)\big) \vee \big(B(0,1) \wedge C(0,2)\big) \vee \big(A(0,2) \wedge C(0,1)\big) \vee \big(B(0,2) \wedge C(0,1)\big)$$

$$= A(0,1) \vee \big(A(1,2) \wedge B(0,1)\big) \vee \big(B(0,1) \wedge C(0,2)\big) \vee \big(A(1,2) \wedge C(0,1)\big) \vee \big(B(1,2) \wedge C(0,1)\big)$$

The probability of the union of the phase two cut sets, Equation 3.34, uses the inclusion-exclusion expansion and is shown in Equation 3.37.

$$P\left(\bigcup_{k=1}^{2} C_{k,2}\right) = P\big(A(0,2)\big) + P\big(B(0,2) \wedge C(0,2)\big) - P\big(A(0,2) \wedge B(0,2) \wedge C(0,2)\big) \qquad \textbf{3.37}$$

The probability of the intersection of the unions of the minimal cut sets from the two phases, Equation 3.36, using the Lower Bound approximation to limit the number of terms, is shown in Equation 3.38.

$$P\left(\bigcup_{k=1}^{2} C_{k,1} \cap \bigcup_{k=1}^{3} C_{k,2}\right) \qquad \textbf{3.38}$$

$$\begin{aligned} &\approx P\big(A(0,1)\big) + P\big(A(1,2) \wedge B(0,1)\big) + P\big(B(0,1) \wedge C(0,2)\big) \\ &+ P\big(A(1,2) \wedge C(0,1)\big) + P\big(B(1,2) \wedge C(0,1)\big) \\ &- \Big(P\big(A(0,1) \wedge B(0,1) \wedge C(0,2)\big) + P\big(A(0,1) \wedge C(0,1) \wedge B(1,2)\big) \\ &+ P\big(B(0,1) \wedge A(1,2) \wedge C(0,2)\big) + P\big(B(0,1) \wedge C(0,1) \wedge A(1,2)\big) \\ &+ P\big(B(0,1) \wedge C(0,1) \wedge A(1,2)\big) + P\big(C(0,1) \wedge A(1,2) \wedge B(1,2)\big)\Big) \end{aligned}$$

The probability of exclusive failure in phase 2, an estimate due to the use of the Lower Bound Approximation in Equation 3.38, is shown in Equation 3.39.

$$Q_2 = P\left(\cup_{k=1}^{2} C_{k,2}\right) - P\left(\left(\cup_{k=1}^{2} C_{k,2}\right) \cap \left(\cup_{k=1}^{3} C_{k,1}\right)\right) \quad \textbf{3.39}$$

$$\approx P\big(A(0,2)\big) + P\big(B(0,2) \wedge C(0,2)\big) - P\big(A(0,2) \wedge B(0,2) \wedge C(0,2)\big)$$
$$- \Big(P\big(A(0,1)\big) + P\big(A(1,2) \wedge B(0,1)\big) + P\big(B(0,1) \wedge C(0,2)\big)$$
$$+ P\big(A(1,2) \wedge C(0,1)\big) + P\big(B(1,2) \wedge C(0,1)\big)$$
$$- \Big(P\big(A(0,1) \wedge B(0,1) \wedge C(0,2)\big) + P\big(A(0,1) \wedge C(0,1) \wedge B(1,2)\big)$$
$$+ P\big(B(0,1) \wedge A(1,2) \wedge C(0,2)\big) + P\big(B(0,1) \wedge C(0,1) \wedge A(1,2)\big)$$
$$+ P\big(B(0,1) \wedge C(0,1) \wedge A(1,2)\big) + P\big(C(0,1) \wedge A(1,2) \wedge B(1,2)\big)\Big)\Big)$$

$$\approx 6.068E-03$$

Equation 3.39 shows that the use of the exact calculation, even with the simple phased mission used in the example and use of the Lower Bound Approximation, contains many terms with resultant computational expense. The same calculation using the approximation method, Equation 3.12, will now be shown to demonstrate the reduction in computational expense and the high accuracy achieved.

Both the failure phase minimal cut sets have superset cut sets in the success phase, therefore $k_1 = 0$ in Equation 3.12. $C_{1,2} = A(0,2)$ and has one superset in the success phase cut sets, $SUP_{1,2} = A(0,1)$, since $A(0,1) = 1$ implies that $A(0,2) = 1$. $C_{2,2} = B(0,2)C(0,2)$ and has two supersets in the success phase cut sets, $SUP_{2,2} = \{B(0,1), C(0,1)\}$, since both $B(0,1) = 1$ and $C(0,1) = 1$ imply that $B(0,2)C(0,2) = 1$. The calculation of the first term from Equation 3.12 is shown in Equation 3.40 and the two parts of the calculation of the second term are shown in Equations 3.41 and 3.42.

$$\sum_{k=1}^{0} P\big(C_{k,2}\big) = 0 \quad \textbf{3.40}$$

$$\sum_{k=1}^{2} P\big(C_{k,2}\big) = P\big(A(0,2)\big) + P\big(B(0,2) \wedge C(0,2)\big) \quad \textbf{3.41}$$

$$\sum_{k=1}^{2} P\big(SUP_{k,2} \cap C_{k,2}\big) = P\big(A(0,2) \wedge A(0,1)\big) + P\left(B(0,2) \wedge C(0,2) \wedge \big(B(0,1) \vee C(0,1)\big)\right) \quad \textbf{3.42}$$

$$= P\big(A(0,1)\big) + P\big(B(0,1) \wedge C(0,2)\big) + P\big(B(0,2) \wedge C(0,1)\big) - P\big(B(0,1) \wedge C(0,1)\big)$$

The calculation for the estimate of the exclusive phase 2 failure probability is shown in Equation 3.43.

$$Q_2 \approx \sum_{k=1}^{0} P\left(C_{k,2}\right) + \sum_{k=1}^{2} \left(P\left(C_{k,2}\right) - P\left(SUP_{k,2} \cap C_{k,2}\right)\right) \quad \textbf{3.43}$$

$$= P\left(A(0,2)\right) + P\left(B(0,2) \wedge C(0,2)\right)$$
$$- \left(P\left(A(0,1)\right) + P\left(B(0,1) \wedge C(0,2)\right) + P\left(B(0,2) \wedge C(0,1)\right)\right.$$
$$\left. - P\left(B(0,1) \wedge C(0,1)\right)\right)$$

$$= 6.207E-03$$

Kohda et al method

The exclusive phase two failure probability will now be calculated through the method given by Kohda et al, Equation 3.19. The equation for the exclusive phase 2 failure probability is shown in Equation 3.44. The success conditions for phase one are calculated through Equation 3.17 and $N_{mps_1} = 1$ since there is only one path set for this phase, giving the success conditions shown in Equation 3.45. The failure conditions for phase 2 are calculated through Equation 3.9 and $N_{mcs_2} = 2$ as there are two cut sets for this phase, giving the failure conditions shown in Equation 3.46. The final calculation of the exclusive phase 2 failure probability, utilising the inclusion-exclusion expansion, is shown in Equation 3.47.

$$Q_2 = P\left(\left(\bigcap_{i=1}^{1} S_i\right) \cap X_2\right) = P\left(S_1 \cap X_2\right) \quad \textbf{3.44}$$

$$S_1 = \bigcup_{j=1}^{1} \bigcap_{l \in P_{1,2}} l(2,\infty) = A(1,\infty) \wedge B(1,\infty) \wedge C(1,\infty) \quad \textbf{3.45}$$

$$X_2 = \bigcup_{k=1}^{2} C_{k,2} = \bigcup_{k=1}^{2} \left(\bigcap_{l \in C_{k,2}} l\right) = A(0,2) \vee B(0,2) \vee C(0,2) \quad \textbf{3.46}$$

$$Q_2 = P\left(\left[A(1,\infty) \wedge B(1,\infty) \wedge C(1,\infty)\right] \wedge \left[A(0,2) \vee \left(B(0,2) \wedge C(0,2)\right)\right]\right) \quad \textbf{3.47}$$

$$= P\left(\left[A(1,2) \wedge B(1,\infty) \wedge C(1,\infty)\right] \vee \left[A(1,\infty) \wedge B(1,2) \wedge C(1,2)\right]\right)$$

$$= P\left(A(1,2)\right) \cdot P\left(B(1,\infty)\right) \cdot P\left(C(1,\infty)\right) + P\left(A(1,\infty)\right) \cdot P\left(B(1,2)\right) \cdot P\left(C(1,2)\right)$$

$$- P\left(A(1,2)\right) \cdot P\left(B(1,2)\right) \cdot P\left(C(1,2)\right)$$

$$= 6.068E-03$$

Example Conclusion

A comparison of the results from the exact methods from Dazhi and Xiaozhong and Kohda et al, shown in Equations 3.39 and 3.47 respectively, shows that the methods agree, giving the same correct value. However it is clear that the final probability calculation from Kohda et al involves far fewer terms and therefore has better computational performance. Dazhi and Xiaozhong's approximation method gives a good, conservative estimate of the unreliability, within 2.5% of the exact value, whilst reducing the computational complexity significantly compared to their exact method. Comparing Koda et al's method, Equation 3.47, with the approximation method from Dazhi and Xiaozhong, Equation 3.43, shows that Kohda et al's method has similar computational expense whilst giving exact results. Kohda et al's approach of directly modelling the success phases using success fault trees and path sets is therefore superior.

3.3.2 BDD based methods

Quantifying the reliability of a system when its structure function is represented by a BDD is far faster and has better computational efficiency than when cut sets and the inclusion-exclusion expansion are used. In a phased mission, once a non-repairable component fails it remains in that failure mode for the remainder of the mission and therefore a basic event for component *A* and failure mode *b* appearing in the phase *j* fault tree is converted into *ite* format as: $ite(A_b(0,j),1,0)$. Whilst in a non-phased non-repairable BDD dependencies can only exist between events on a path belonging to different failure modes of the same component, in a phased mission BDD, dependencies also occur between events in different phases from the same component. The BDD based methods described in this section use different strategies for resolving these dependencies between phases, failure modes, or both, during the BDD construction from the mission fault tree and the probability evaluation procedure.

3.3.2.1 Systems with single failure mode components

Rauzy [14] presented the first work on the use of the BDD method for the quantification of non-phased mission fault trees. Zang, Sun and Trivedi [31] extended this method for the analysis of systems operating in phased missions that have single failure mode components.

BDD Construction

Zang et al provide a BDD build procedure, which outputs a resultant BDD node from a logical operation performed between two other nodes. The process depends on the variable order scheme of which two possibilities are considered – Forward Phase-Dependent Operation (PDO) and Backward PDO. The ordering schemes both order variables by component first and only vary in the ordering of variables belonging to the same component but in different phases. In the Forward PDO, the ordering of the variables matches the phase order such that earlier phases appear higher in the BDD, whereas with Backward PDO it is the reverse. The BDD computation algorithm, modified from that of Rauzy, uses phase algebra to produce a smaller BDD.

If the two nodes in the computation belong to different components then the standard BDD computation rules, given by Equations 2.14 to 2.17, are used. Otherwise, the modified BDD computation rules shown in Equation 3.48 are used, where *x* and *y* are variables belonging to the same component and the variable ordering scheme is the Forward PDO. Comparing Equation 3.48 to Equations 2.14 to 2.17 shows that the failure child of the resultant node is computed from $F1 \oplus G1$ instead of $F1 \oplus G$ when $index(x) < index(y)$. This is because, when *x* and *y* belong to the same component and the Forward PDO is used, x=1 implies y=1, since if a non-repairable component is failed in a phase it will also be failed in later phases.

$$ite(x, F1, F0) \oplus ite(y, G1, G0) = \begin{cases} ite(x, F1 \oplus G1, F0 \oplus G0) & index(x) = index(y) \\ ite(x, F1 \oplus G1, F0 \oplus G) & index(x) < index(y) \end{cases} \quad \textbf{3.48}$$

An example of the BDD size reduction is shown by comparing Figure 3.3a, a BDD constructed through Equations 2.14 to 2.17 and Figure 3.3b, a smaller BDD representing the same structure function that was constructed through Equation 3.48.

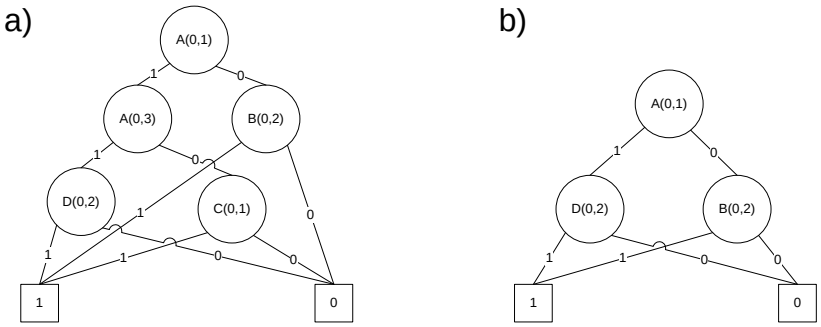


Figure 3.3 – Reduction in BDD size through modified BDD algorithm from Zang et al.

The computation rules corresponding to Equation 3.48 but for a Backward PDO ordering are shown in Equation 3.49. Zang et al note that the BDD is smaller when the Backward PDO ordering is used.

$$ite(x,F1,F0)\oplus ite(y,G1,G0) = \begin{cases} ite(x,F1\oplus G1,F0\oplus G0) & index(x)=index(y) \\ ite(x,F1\oplus G,F0\oplus G0) & index(x)<index(y) \end{cases} \quad \textbf{3.49}$$

BDD Evaluation

Once the phased mission fault tree has been converted into a BDD using the procedure outlined above, it must be evaluated to find the top event probability. Evaluating the top event probability from the BDD is more complex than in the single phased case where a node's variables are independent of the variables in its children. A BDD node in a phased mission BDD may have child nodes containing dependant variables and it is no longer correct to derive the node probability as the sum of the products of its variables failure and success events and the corresponding child nodes probabilities. To resolve the dependencies in the BDD and determine the correct node probability, Zang et al provide an evaluation procedure that is based upon the phase algebra of [30] shown in Equations 3.21 to 3.26. The recursive procedures, specified for a BDD built with a Backward PDO ordering, shown in Equations 3.51 and 3.52 are used when variables *x* and *y* belong to different components and the same component respectively, where node *F* is as was defined in Equation 2.14, i.e. $F = ite(x, F_{x=1}, F_{x=0}) = ite(x, F1, F0)$, *F1* is defined as shown in Equation 3.50 and $cp(x_i)$ is the component of the event represented by the Boolean variable x_i.

$$F1 = ite(y, F1_{x=1}, F1_{x=0}) = ite(x, H1, H0) \quad \textbf{3.50}$$

$$P(F) = P(F1) + (1-P(x))(P(F0) - P(F1)) \text{ , if } cp(x) \neq cp(y). \quad \textbf{3.51}$$

$$P(F) = P(F1) + (1-P(x))(P(F0) - P(H0)) \text{ , if } cp(x) = cp(y). \quad \textbf{3.52}$$

The method allows the probabilities to be cached for each BDD node, avoiding repeating calculations for nodes with multiple parents, and has far better computational efficiency than the earlier non-BDD phased mission methods. It can also be used to evaluate an exclusive phase failure fault tree and therefore derive the probability that mission failure occurs in a specific phase.

Alternative method

La Band and Andrews [32][33] presented another method and suggest the use of the BDD technique for better evaluation efficiency. Unlike the other Boolean phase algebra based methods, the method involves replacing component failure events in each phase fault tree by an OR gate with inputs of events representing the failure of the component in each phase of the mission up to and including that phase, as in the EZ method. The Boolean phase algebra rules from La Band and Andrews are shown in Equations 3.53 to 3.59. However, these rules do not resolve dependencies in many cases, for example none of the rules resolve the dependencies between the events $\overline{A}(0,i)$ and $A(0,j)$.

$A(i-1,i) \wedge A(j-1,j) = 0$, where $i < j$. **3.53**

$A(i-1,i) \wedge A(i-1,j) = A(i-1,i)$, where $i < j$. **3.54**

$\overline{A}(i-1,i) \wedge A(i-1,i) = 0$ **3.55**

$\overline{A}(i-1,i) \wedge A(j-1,j) = A(j-1,j)$, where $i < j$. **3.56**

$\overline{A}(i-1,i) \wedge A(i-1,j) = A(i,j)$, where $i < j$. **3.57**

$\overline{A}(i-1,i) \wedge \overline{A}(i,i+1) \wedge \ldots \wedge \overline{A}(j-1,j) = \overline{A}(i-1,j)$, where $i < j$. **3.58**

$A(i-1,i) \vee A(i,i+1) \vee \ldots \vee A(j-1,j) = A(i-1,j)$, where $i < j$. **3.59**

The use of the BDD method, using the BDD computation algorithm from [14], is suggested for more efficient analysis than the use of the implicants and inclusion-exclusion expansion. The replacing of component failure events with the union of their individual phase events and a non-phased mission specific BDD algorithm result in far larger BDDs than when the methods given in [31] are used. The quantification algorithm is to sum the probabilities of each path to a terminal one node, a procedure that does not permit probabilities to be cached within BDD nodes and this together with the larger BDD size results in higher computational expense than the method from Zang et al.

3.3.2.2 Systems with multiple failure mode components

The methods discussed so far have been concerned with the analysis of systems with single failure mode components. Since many real world systems both operate in phased missions and contain components with multiple failure modes, the development of methods capable of analysing them is an important topic. Zang, Wang, Sun and Trivedi [34] extended the BDD algorithms from [14] to analysis of systems containing components with multiple competing failure modes. They utilised Boolean algebra from Caldarola [35] to deal with dependencies between variables belonging to the same component. Tang and Dugan [36] combined the phased mission BDD techniques from Zang et al [31] with the multiple competing failure mode BDD techniques from Zang et al [34], resulting in a BDD method capable of analysing phased mission systems containing multiple competing failure modes.

Construction of the BDD

In this section, the process of constructing a BDD from a phased mission fault tree according to the method given by Tang and Dugan will be explained. Two sets of Boolean phase multi-failure mode algebra are given, the first set dealing with dependencies between events belonging to different failure modes of the same component, Equations 3.60 to 3.64, and the second dealing with dependencies between events belonging to the

same failure mode, Equations 3.65 to 3.70, where A_b refers to the occurrence event for failure mode b of component A.

$A_a(0,i) \wedge A_b(0,j) = 0$, where $i \leq j$ and $a \neq b$. **3.60**

$\overline{A_a}(0,i) \wedge A_b(0,j) = A_b(0,j)$, where $i \leq j$ and $a \neq b$. **3.61**

$A_a(0,i) \wedge \overline{A_b}(0,j) = A_a(0,i)$, where $i \leq j$ and $a \neq b$. **3.62**

$A_a(0,i) \vee \overline{A_b}(0,j) = \overline{A_a}(0,j)$, where $i \leq j$ and $a \neq b$. **3.63**

$\overline{A_a}(0,i) \vee A_b(0,j) = \overline{A_a}(0,i)$, where $i \leq j$ and $a \neq b$. **3.64**

$A_a(0,i) \wedge A_a(0,j) = A_a(0,i)$, where $i < j$. **3.65**

$\overline{A_a}(0,i) \wedge \overline{A_a}(0,j) = \overline{A_a}(0,j)$, where $i < j$. **3.66**

$A_a(0,i) \wedge \overline{A_a}(0,j) = 0$, where $i < j$. **3.67**

$\overline{A_a}(0,i) \vee \overline{A_a}(0,j) = \overline{A_a}(0,i)$, where $i < j$. **3.68**

$A_a(0,i \vee A_a(0,j) = A_a(0,j)$, where $i < j$. **3.69**

$\overline{A_a}(0,i) \vee A_a(0,j) = 1$, where $i < j$. **3.70**

The combinations $\overline{A_a}(0,i) \wedge \overline{A_b}(0,j)$ and $A_a(0,i) \vee A_b(0,j)$ cannot be resolved into a single event and their respective probabilities are found through Equations 3.71 and 3.72.

$$P\left(\overline{A_a}(0,i) \wedge \overline{A_b}(0,j)\right) = 1 - P\left(A_a(0,i)\right) - P\left(A_b(0,j)\right) \quad \textbf{3.71}$$

$$P\left(A_a(0,i) \vee A_b(0,j)\right) = P\left(A_a(0,i)\right) + P\left(A_b(0,j)\right) \quad \textbf{3.72}$$

In addition, as with the rules from Somani and Trivedi, the combination $\bar{A}(0,i) \wedge A(0,j)$ is not resolved by these rules and its probability should be calculated through Equation 3.27, as before.

The variable ordering for the BDD is stringent from the view of implementation, meaning that the BDD construction and evaluation algorithms are developed with the assumption that it is used. The variable ordering scheme is for variables to be ordered first by component, then by backward phase (such that later phases appear higher) and finally by failure mode. The resultant node from a logical operation between two nodes, *F* and *G*,

shown in equations 3.73 and 3.74 respectively, is determined through Equation 3.75, where L1 and L0 are defined in Equations 3.76 and 3.77 respectively.

$$F = ite(x, F_{x=1}, F_{x=0}) = ite(x, F1, F0) \quad \textbf{3.73}$$

$$G = ite(y, G_{y=1}, G_{y=0}) = ite(y, G1, G0) \quad \textbf{3.74}$$

$$ite(x, F1, F0) \oplus ite(y, G1, G0) = \quad \textbf{3.75}$$

$ite(x, F1 \oplus G1, F0 \oplus G0)$, if $index(x) = index(y)$

$ite(x, F1 \oplus L1, F0 \oplus G)$, if $index(x) < index(y), cp(x) = cp(y), fm(x) \neq fm(y)$

$ite(x, F1 \oplus G, F0 \oplus L0)$, if $index(x) < index(y), cp(x) = cp(y), fm(x) = fm(y)$

$ite(x, F1 \oplus G, F0 \oplus G)$, *otherwise*

When *x* and *y* are from the same component but a different failure mode then *x=1* implies *y=0* since only one mutually exclusive failure mode can occur at any time, justifying *L1* in Equation 3.75. When *x* and *y* are from the same component and the same failure mode, but *x* is ordered lower then it must be from a later phase (due to the ordering strategy) and therefore *x=0* implies *y=0*, justifying *L0* in Equation 3.75.

$$L1 = G0_{x=1} \quad \textbf{3.76}$$

$$L0 = G0_{x=0} \quad \textbf{3.77}$$

Tang and Duggan [36] give Equation 3.78 as the procedure for determining *L1* from node *G*. In Equation 3.78, if the variable of *F0* belongs to same component as *x* then *F0* is extended as *F* in Equation 3.78 until reaching sub-expressions which do not have variables that belong to the same component as *x*.

if $cp(x) = cp(y), fm(x) \neq fm(y)$, **3.78**

$$\begin{aligned}(G)_{x=1} &= (y \cdot G1 + \bar{y} \cdot G0)_{x=1} \\ &= 0 \cdot G1_{x=1} + 1 \cdot G0_{x=1} \\ &= L_1\end{aligned}$$

Problems identified in construction method

Some problems with the BDD construction method given by Tang and Dugan [36] were identified during its review and these are explained in this section.

Equation 3.78 is an incorrect procedure for finding *L1* and this will now be shown through an example. The BDD shown in Figure 3.4a represents node *G*, $ite\left(A_1(0,2),1,ite\left(A_2(0,1),1,ite(A_3(0,1),1,0)\right)\right)$ in *ite* format. Therefore *y*, the variable of node *G*, is $A_1(0,2)$. In this example, *L1* is to be formed as *G* for $x=1$, where *x* is the variable $A_2(0,2)$. The correct BDD for *L1* is shown in Figure 3.4b, $ite(A_1(0,1),1,0)$ in *ite* format.

From Tang and Duggan, since *x* is a variable indexed lower than *y* in the ordering as it is from a later phase, and is also from a different failure mode, Equation 3.78 should be applied to determine *G* when $x=1$, i.e. to determine *L1*. Now according to Equation 3.78, *L1* is determined by following the success children of *G* until a sub-expression is reached that does not contain variables that belong to the same component as variable *x*, i.e. component *A*. Since all variables in *G* belong to component *A*, according to Equation 3.78, *L1* is represented by the BDD shown in Figure 3.4c, a terminal 0 node. It is therefore clear that Equation 3.78 is not the correct approach to finding *L1* as it does not give the correct result of $ite(A_1(0,1),1,0)$. The correct approach is to simply replace all nodes in *G* that belong to the same component as *x* but belong to a different failure mode with their success child node.

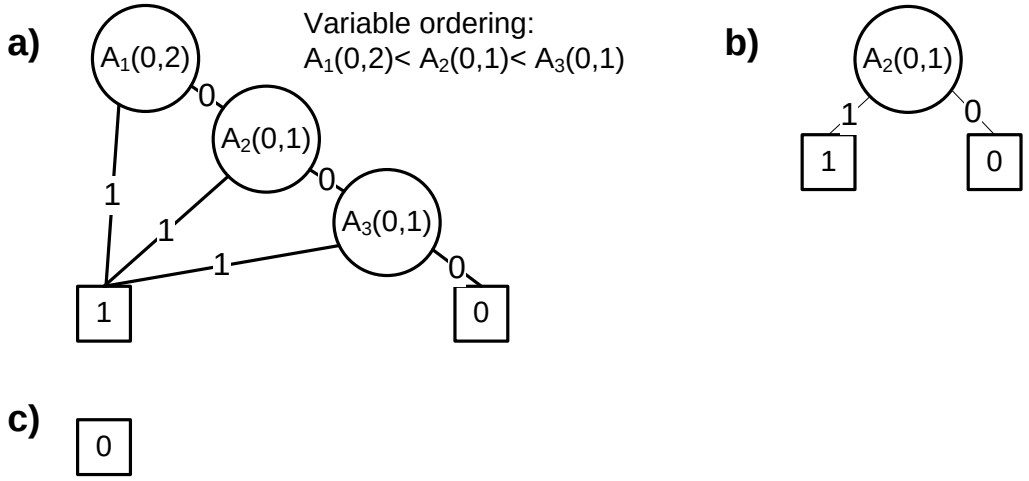


Figure 3.4 – BDD construction example 1.

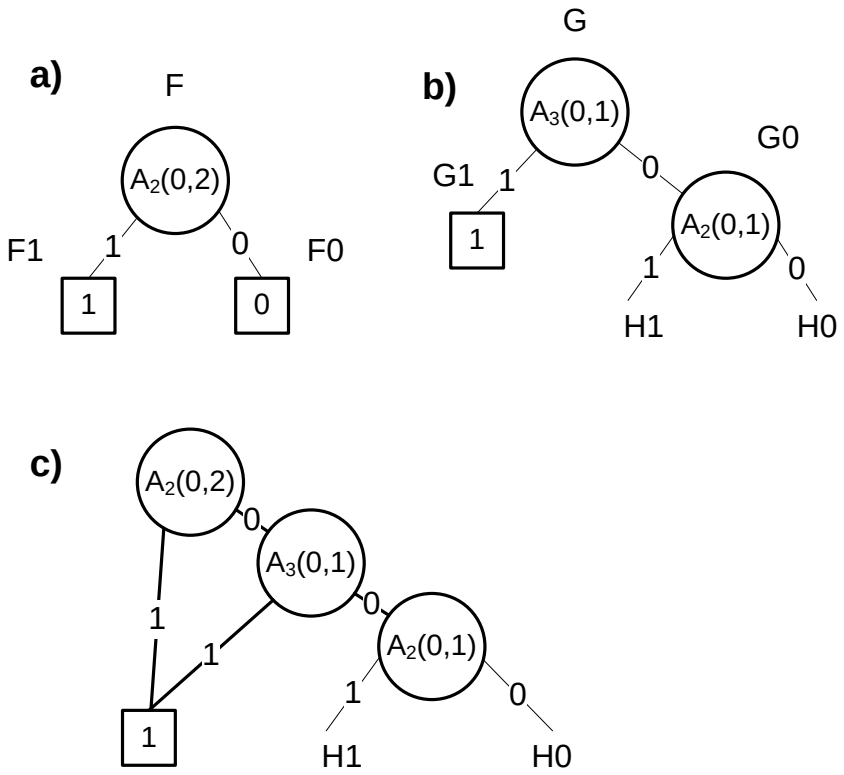


Figure 3.5 – BDD construction example 2.

In addition, the BDD computation algorithm in Equation 3.75 does not minimise BDD size. From Equation 3.75, $ite(x, F1, F0) \oplus ite(y, G1, G0) = ite(x, F1 \oplus L1, F0 \oplus G)$ when $index(x) < index(y), cp(x) = cp(y), fm(x) \neq fm(y)$, as is the case when *F* and *G* are the BDDs shown in Figure 3.5a and Figure 3.5b respectively, or $ite(A_2(0,2),1,0)$ and $ite(A_3(0,1),1,ite(A_2(0,1),1,0))$ in *ite* format. The BDD shown in Figure 3.5c, $ite(A_2(0,2),1,ite(A_3(0,1),1,ite(A_2(0,1),H1,H0)))$ in *ite* format, therefore results from a Boolean OR operation between *F* and *G*. However, this resultant node is not minimal since the path to node H1 never occurs because if $A_2(0,2) = 0$ then it is certain that $A_2(0,1) = 0$ and therefore the node labelled with variable $A_2(0,1)$ in Figure 3.5c, $ite(A_2(0,1),H1,H0))$ in *ite* format, can be replaced by *H0*. Since *H1* could be a large BDD, the non minimised BDD could consume a lot of unnecessary memory.

BDD Probability Evaluation

The next section gives the methods and equations used to evaluate the probability from the BDD.

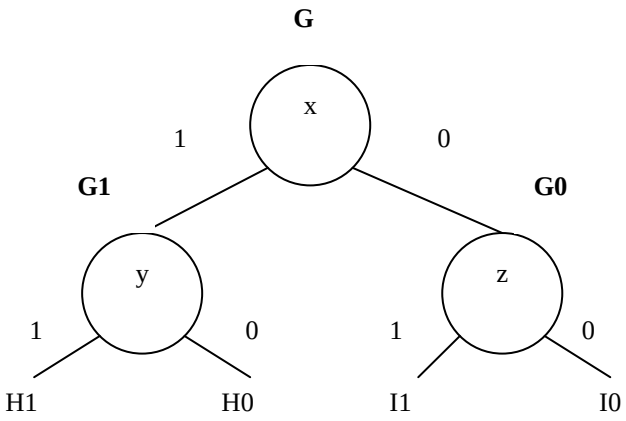


Figure 3.6 – Labels for a BDD node's relative children

The probability for BDD node G, depicted in Figure 3.6 along with its relatively positioned child nodes *G1*, *G0*, *H1*, *H0*, *I1* and *I0*, is evaluated through one of Equations 3.79 to 3.82. The choice of Equations 3.79 to 3.82 depends on which of the cases shown in Table 3.3 describes the relationship between the variables *x*, *y* and *z* from nodes *G*, *G1* and *G0* respectively. *K2* in Equations 3.80 and 3.81 is defined by Equation 3.83, where W_1, $W_{2,\ldots,}$ W_n are variables with ascending order in *I0* that all belong to the same component as variable *x*.

$$P(G1=1)+P(G0=1)-P(H0=1)+P(x=1)\cdot\big(P(H0=1)-P(G0=1)\big), \text{ if case 1.} \quad \textbf{3.79}$$

$$P(G1=1)+P(G0=1)-P(H0=1)+P(x=1)\cdot\big(P(H0=1)-P(K2=1)\big), \text{ if case 2.} \quad \textbf{3.80}$$

$$P(G0=1)+P(x=1)\cdot\big(P(G1=1)-P(K2=1)\big), \text{ if case 3.} \quad \textbf{3.81}$$

$$P(G0=1)+P(x=1)\cdot\big(P(G1=1)-P(G0=1)\big), \text{ if case 4.} \quad \textbf{3.82}$$

Table 3.3 - BDD node probability evaluation cases corresponding to Equations 3.79 to 3.82.

Case	Conditions
1	$cp(x)=cp(y),\ fm(x)=fm(y),\ pn(x)\neq pn(y),\ cp(x)\neq cp(z)$
2	$cp(x)=cp(y),\ fm(x)=fm(y),\ pn(x)\neq pn(y),\ cp(x)=cp(z),\ fm(x)\neq fm(z)$
3	$cp(x)\neq cp(y),\ cp(x)=cp(z),\ fm(x)\neq fm(z)$
4	$cp(x)\neq cp(y),\ cp(x)\neq cp(z)$

Note that in the paper from Tang and Dugan, cases 2 and 3 are given with $fm(x) = fm(z)$ instead of the intended $fm(x) \neq fm(z)$ that is shown in Table 3.3. Cases 2 and 3 should have $fm(x) \neq fm(z)$ since in these cases *x* and *z* belong to the same component and as Tang and Duggan state, the 0 branch always links two variables that belong to different failure modes of the same component, or belong to different components.

$$K2 = I0_{x=1} = I0_{W_1=W_2=..=W_n=0} \qquad \textbf{3.83}$$

Comparison of performance of evaluation algorithm with single failure mode equivalent

The performance of the BDD evaluation algorithm described above will now be compared with the earlier BDD method developed for the analysis of single failure mode phased missions by Zang, Sun and Trivedi [34]. The most complex case for the calculation of the probability of a BDD node for each of the methods is described below:

Zang et al method when variables *x* and *y* belong to the same component:

1. Calculate probability of the probability of the failure event for variable *x*.
2. Get probabilities from nodes *G1* and *G0*.
3. Check if variable *x* and variable *y* belong to the same component.
4. Get probability from node *H0*.
5. Calculate probability from these values.

Tang and Duggan method when variables *x* and *y* belong to the same component, failure mode and variables *x* and *z* belong to different components:

1. Calculate probability of the probability of the failure event for variable *x*.
2. Check if variable *x* and variable *y* belong to the same component.
3. Check if variable *x* and variable *y* belong to the same failure mode.
4. Check if variable *x* and variable *z* belong to the same component.
5. Get probabilities from nodes *G1* and *G0*.
6. Get probability from node *H0*.
7. Traverse down BDD along success children, starting at node *I0*, comparing nodes variable with *x* until node whose variable belongs to a different component is encountered.
8. Calculate probability from these values.

As shown above the method from Tang and Duggan contains all the steps from the method by Zang et al and the addition of three more steps, including step 7 which is an iterative rule that may involve a large number of variable comparisons. Thus, evaluating the probability of a node in Tang and Duggan's method can involve far greater computational expense.

Errors in BDD evaluation method

Some problems with the BDD evaluation algorithms given by Tang and Duggan were identified during the review of their method, which will be discussed in this section. To show that the BDD evaluation algorithm is incorrect, it is sufficient to show that it gives an incorrect value for the probability of a node that can be formed through their BDD construction algorithm. For simplicity, assume that *H1* and *HO* in Figure 3.5c contain no variables belonging to component *A*. The probability for the BDD in Figure 3.5c is then given by Equation 3.84.

$$P(Top) = P(A_2(0,2)) + P(A_3(0,1)) + (1 - P(A_2(0,2)) - P(A_3(0,1))) \cdot P(H_0) \quad \textbf{3.84}$$

A value for the probability of the BDD will now be calculated through the procedure given by Tang and Duggan. The top node of Figure 3.5c is to be evaluated according to case 3 from Table 3.3, through Equation 3.81, as shown in Equation 3.85 where *U* is its success child node. Node *U* is also evaluated according to case 3 from Table 3.3, through Equation 3.81, as shown in Equation 3.86, where *V* is its success child node. Node *V* is evaluated according to case 4 from Table 3.3, through Equation 3.82, as shown in Equation 3.87.

$$P(Top) = P(U) + P(A_2(0,2)) \cdot (1 - P(H_0)) \quad \textbf{3.85}$$

$$P(U) = P(V) + P(A_3(0,1)) \cdot (1 - P(H_0)) \quad \textbf{3.86}$$

$$P(V) = P(H_0) + P(A_2(0,1)) \cdot (P(H_1) - P(H_0)) \quad \textbf{3.87}$$

This example shows that the Tang and Duggan evaluation procedure is clearly incorrect since using it to determine the probability of the top node in Figure 3.5c, gives a value that depends on H_1, whereas the correct value, shown in Equation 3.84, does not.

3.3.3 Summary and Conclusions

A large number of methods have been developed for analysing the reliability of non-repairable phased missions. The earliest methods relied on transforming a phased mission into an equivalent non-phased mission, resulting in computationally expensive analysis. Methods using Boolean phase algebras were then introduced and give better computational efficiency, whilst the most recent, and most computationally efficient, methods use the BDD technique. The Boolean phase algebra based methods offer clear advantages over the earlier methods; the computational efficiency is far higher and most are capable of directly evaluating exclusive phase failure probabilities directly. A wide range of different Boolean phase algebras have been developed. All except the earliest method are able to resolve combinations that include success, in addition to failure, events, allowing them to be used with methods that can find probability of exclusive phase failure in a specific phase. The only algebra able to resolve all resolvable combinations for single failure mode components, was developed by Kohda et al and also has the benefit of being succinct - consisting of just three, easily implemented rules. The early Boolean phase algebra methods are still computationally expensive, since the inclusion-exclusion expansion must be used in each of the probability calculations. The full expansion of all terms is particularly important with the method from Kohda et al when calculating the probability of exclusive phase failure or mission success, since the component success events are unlikely to have the low probabilities that would allow

the approximate expansions based on the rare event assumption to be used. However, the direct modelling the success of earlier phases in the method by Kohda et al results in lower computational expense than the indirect approach taken in Dazhi and Xiaozhong.

The use of the BDD technique in the later methods reduces the computational expense significantly. Zang et al presented a method utilising phased mission specific BDD conversion and evaluation techniques to offer very fast analysis of phased missions for systems with single failure mode components. Tang and Dugan presented a method and algebra for dealing with multiple failure mode component phased mission components, although there are some problems with the methods presented in their research. The evaluation of the BDD in this method is not as fast as the methods for single failure mode component based systems given by Zang et al, due to the potential distance between a node and the nodes with cached probabilities that are used in its probability calculation. The fast evaluation of phased missions for systems with multiple failure mode components is therefore an area for further research.

4 New methods for the analysis of non-repairable phased missions

4.1 Introduction

The literature review presented a wide range of methods developed for the analysis of non-repairable phased missions. The most recent methods use Boolean phase algebra together with the Binary Decision Diagram technique to resolve dependencies in event combinations and provide computationally efficient analysis.

The method developed by Tang and Dugan [36] is the first for the analysis of phased missions that includes components with multiple failure modes. Systems with multiple failure mode components are likely to become more common as systems become more sophisticated, more integrated, and are required to carry out an increasing number of tasks. This increased sophistication increases the likelihood that the mode of failure for a component will affect the system reliability in different ways, thus increasing the importance of modelling multiple failure modes. Unfortunately, the method for the analysis of this type of system presented by Tang et al contains a number of errors that can lead to incorrect results, as was shown in section 3.3.2.2 of the literature review.

The ability to analyse the reliability of a system in real time, where its reliability prediction is constantly updated to reflect its current situation, has recently received attention [37] and is likely to be of increasing significance as systems for which automation in decision making is important, such as UAVs [38], become more common. Recently, importance measures for phased missions have been developed by Andrews [2], as well as those developed in chapter 8 of this thesis. Both real time and importance measure analysis often require the re-evaluation of the probability of the same reliability structure but with different component reliabilities and phase durations, and, particularly for real time analysis, require that it is performed in very small time scales. Whilst the method for phased mission systems with single failure mode components by Zang, Sun and Trivedi [31] has similar efficiency to the original BDD methods from Rauzy, the method from Tang et al does not perform so well. This is due to the potential for a substantial increase in the number of node variable comparisons and traversal distance during the probability evaluation procedure as was shown in section 3.3.2.2.

This chapter presents a new method that has been developed to address the problems outlined above. It accurately analyses the reliability of phased mission systems with multiple failure mode components and offers increased efficiency compared to the method from Tang and Duggan – particularly when probability re-evaluations of the same reliability structure are required. A software tool that implements this method has also been developed and is described in this chapter.

4.2 New Binary Decision Diagram based method

The latest research, such as from Zang et al [31], La Band and Andrews [32] and Tang et al [36] have all used the Binary Decision Diagram technique as it offers far better computational efficiency than the earlier methods that relied on the derivation of cut sets and the use of the inclusion-exclusion expansion. In these techniques the

phased mission fault tree is first converted into its BDD equivalent and then a probability evaluation is performed. The algorithms for the construction and quantification of the BDDs given by Zang et al and Tang et al were studied to determine where any improvements could be made. The method from Zang et al, suitable for the analysis of systems containing single failure mode components, was found to be very efficient, with a lower efficiency only when forward rather than backward phase orderings are used (i.e. where variables belonging to earlier phases appear higher in the BDD) identified as an area for improvement. However, a number of possible improvements were identified with the method for the more complex case of systems containing multiple failure mode components by Tang et al. As was shown in the literature review, the method contains errors that lead to incorrect results for the probability and additionally, the BDD probability evaluation has a high computational expense due to the significant node traversal and node variable comparisons involved in the resolution of the phase and failure mode dependencies.

Common to both methods is that the conversion of node variables into events, their probability evaluation, and the resolution of dependencies are integrated together in the calculation of a node's probability. This means that if the probability evaluation is repeated, with different phase durations for example, then all three steps must be repeated, entailing unnecessary computation when only the probability of the events has changed. Repeated calculation of the top event probability from the same mission tree is required for certain types of analysis such as the calculation of component importance measures, see chapters 7 and 8, and real time analysis where it is constantly updated as the mission progresses (and hence phase durations change) and monitored components fail.

Improvements have been developed for all the areas identified. An additional procedure has been developed that, when added to the fault tree to the BDD computation algorithm from Tang et al, results in a BDD of minimal size with both backward and forward phase orderings. For the analysis of systems with multiple failure mode components, an alternative to the method from Tang et al has been developed. It uses a different variable ordering scheme and introduces an additional step between the construction of the BDD and the probability evaluation. This new method gives correct results for this type of system and does so through a procedure that offers lower computational expense. It is particularly advantageous, in terms of increased efficiency, when multiple probability evaluations of the same mission fault tree are required, such as when carrying out real time or importance measure analysis.

4.2.1 BDD Construction

In this work the methodology for transforming the mission fault tree into a BDD is similar to the approach taken by Tang et al but with some changes that increase solution efficiency.

4.2.1.1 Variable Ordering

The method requires the use of a variable ordering that has some restrictions that are stringent from the view of implementation. The ordering of the variables within these restrictions is not considered in this paper although it can have a large effect on BDD size and solution efficiency. Finding the optimum ordering is known to be an non-deterministic polymonial-time (NP) hard problem and has been studied for the case of phased missions with multiple failure mode components by Mo [39]. Each of the BDD nodes will be labelled with a variable of the

form $I_j(0, p)$ and the ordering between two nodes is determined from a comparison between the component, failure mode and phase of their respective variables. The ordering restriction is that nodes are ordered first by component, then by failure mode and then by phase number (note that this differs from the ordering used by Tang and Dugan [36] where nodes are ordered by component, then by phase number and then by failure mode). An example of a valid ordering for a system consisting of two components, *A* and *B*, each with two failure modes, 1 and 2, when operating in a two phased mission would be $A_1(0,1) < A_1(0,2) < A_2(0,1) < A_2(0,2) < B_1(0,1) < B_1(0,2) < B_2(0,1) < B_2(0,2)$.

Zang et al [31] showed that their method results in a smaller BDD when used with a backward phase variable ordering (where later phases have a lower index) than a forward phase ordering. Since Tang and Dugan's method is partly based upon that method, it also performs better with that ordering. The method presented in this paper gives greater flexibility however and generates an equally sized BDD with either choice. The forward phase ordering will be used in the descriptions given in this paper.

4.2.1.2 Node Computation

The size of the BDD resulting from the computation algorithm given by Tang and Dugan is often larger than necessary, due to the inclusion of paths containing mutually exclusive events. For example, Figure 4.1 shows a phased mission and Figure 4.2 its BDD computed according to the method given by Tang and Dugan.

The BDD node labelled $A_1(0,1)$ could be replaced by the terminal 0 node without loss of information, since it is not possible for the failure mode to occur in the first phase if it did not occur in the first two phases (i.e. the path to the node's terminal 1 child will always have a probability of 0). The BDD therefore contains 33% more *ite* nodes than is optimal. Compared to the algorithm used by Tang and Dugan, the node computation algorithm used in the new method is an improvement as it results in the minimally sized BDD for a given variable ordering.

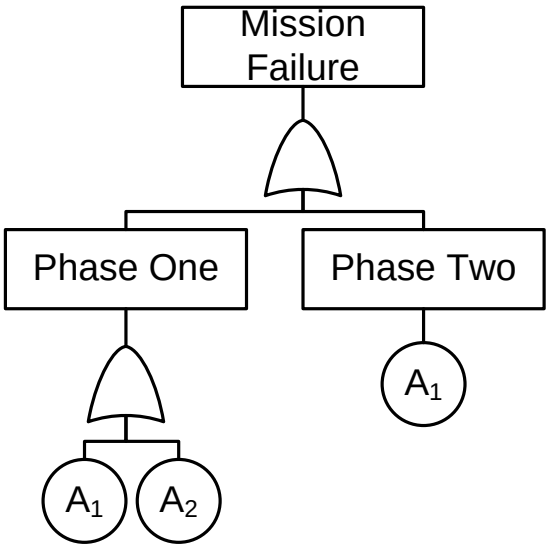


Figure 4.1 - Fault tree for a phased mission system.

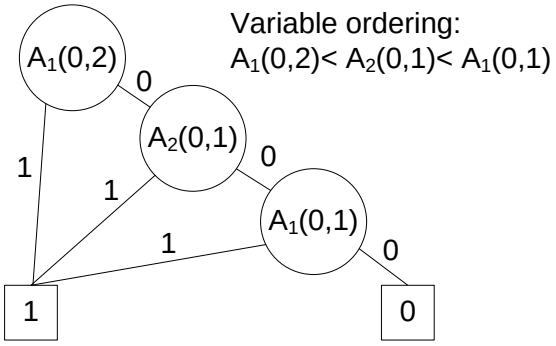


Figure 4.2 - BDD calculated for phased mission shown in Figure 4.1 through Tang and Dugan's method.

The BDD is constructed from the mission fault tree, as before, by converting the fault tree basic events into BDD nodes and then applying Equation 2.15. with the Boolean logical operation of each gate on its inputs. A non-repairable component remains in a failure mode for the remainder of the mission upon entry and therefore a basic event for component *I* and failure mode *j* appearing in the phase *p* fault tree is converted into *ite* format as: $ite\big(I_j(0,p),1,0\big)$.

Changes to the method used to calculate the output node from a Boolean logical operation between two nodes *F* and *G*, as described by Equations 2.14-2.17 for the non-phased mission case, are introduced here and result in a BDD that is optimised for efficient solution. The calculation of the success child of the resultant node, node *V*, remains the same as in the non-phased mission case, using Equation 2.17. The calculation of the failure child of the resultant node, node *U*, also remains the same, using Equation 2.16 when the index of the variables of nodes *F* and *G*, i.e. *x* and *y*, are equal. However, if variable *x* has a lower index than variable *y*, dependencies may exist between variables *x* and *y*, and new rules are used to calculate node *U*, replacing the rule given in Equation 2.16.

The calculation of node *U* when variable *x* has a lower index than variable y, will now be described in detail. The relationship between the variables of nodes *F* and *G*, *x* and *y*, determine which of the exhaustive set of rule conditions is met and accordingly, which of the rules from Table 4.1 to apply in the computation of node *U*. The reasoning behind each rule from Table 4.1 is now explained:

Rule 1: If *x* and *y* are variables of the same failure mode of the same component, then the phase of *x* must be earlier than or equal to the phase of *y* (due to the variable ordering) and *x*=1 implies *y*=1 since the failure mode is non-repairable. This implies that the resultant node's failure child, node *U*, should be formed from the computation of the failure children of node *U* and *V* as shown in rule 1 of Table 4.1.

Table 4.1 – Rules for computing nodes *U* and *V*.

Rule	U and V computation	Condition
1	$U = F1 \oplus G1$	If *x* and *y* are variables belonging to the same failure mode of the same component.
2	$U = F1 \oplus G0_{x=1}$ where $G0_{x=1}$ is equal to *G0* if *G0*'s variable belongs to a different component to *x*, otherwise it is the first node with a variable belonging to a different component encountered during a traversal down the success children of the BDD starting from *G0*.	If *x* and *y* are variables belonging to different failure modes of the same component.
3	$U = F1 \oplus G$	Otherwise.

Rule 2: If *x* and *y* are variables belonging to the same component but different failure modes then *x*=1 implies that *y*, and any other variable belonging to a different failure mode of the same component as *x*, equals 0. This is true since a component's failure modes are mutually exclusive. The failure child of the resultant node, U, is therefore computed from the failure child of node *F*, *F1*, and the first success child of node *G* which has a variable not belonging to the same component as *x*, $G0_{x=1}$. The recursive nature of finding $G0_{x=1}$ through the success child traversal is shown by Equation 4.1 where $H_{x=1}$ is determined until its variable belongs to a different component to x at which point it becomes equal to $G0_{x=1}$.

$$H_{x=1} = \begin{cases} (w.H1 + \overline{w}.H0)_{x=1} & if\ w\ belongs\ to\ the\ same\ component\ as\ x. \\ G0_{x=1} & otherwise. \end{cases} \quad \textbf{4.1}$$

$where\ H = ite(w, H1, H0)\ and\ initially\ H = G.$

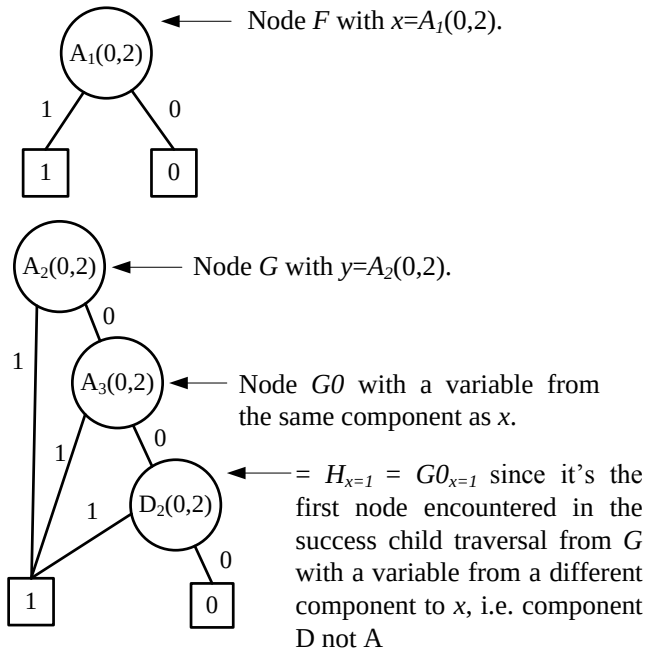


Figure 4.3 - Example application of Equation 4.1.

The procedure to determine $G0_{x=1}$ is illustrated by applying Equation 4.1 in the computation between the nodes shown in Figure 4.3:

$F = ite(A_1(0,2), 1, 0)$

$G = ite\left(A_2(0,2), 1, ite\left(A_3(0,2), 1, ite(D_2(0,2), 1, 0)\right)\right)$

To determine $G0_{x=1}$, Equation 4.1 is applied. As x=1, i.e. $A_1(0,2)=1$, then $A_j(0,p)=0$ for all $j \neq 1$. Initially H=G hence w=$A_2(0,2)$, H1=1, $H0 = ite\left(A_3(0,2), 1, ite(D_2(0,2), 1, 0)\right)$. Applying Equation 4.1 gives:

$$H_{x=1} = (H0)_{x=1} = \left(ite\left(A_3(0,2), 1, ite(D_2(0,2), 1, 0)\right)\right)_{x=1}$$

The variable of H, $A_3(0,2)$, belongs to the same component as x, hence Equation 4.1 must be applied again with w= $A_3(0,2)$, H1=1, $H0 = ite(D_2(0,2), 1, 0)$:

$$H_{x=1} = (H0)_{x=1} = \left(ite(D_2(0,2), 1, 0)\right)_{x=1}$$

In this case the variable of H, $D_2(0,2)$, belongs to a different component to x and hence $G0_{x=1} = ite(D_2(0,2), 1, 0)$.

If *G0* in the Figure 4.3 example had a variable from a different component to *x*, such as $C_1(0,1)$, then $G0_{x=1}$ would instead be G0.

Rule 3: In all other cases, *x* and *y* belong to different components and are independent, therefore *U* is calculated according to Equation 2.16.

A new reduction rule introduced in this paper, is for the case where *x* belongs to the same failure mode as node *V*'s variable, defined here as variable *z*, such that *x*=1 implies that *z*=1, and $V_{z=1}$ and *U* are the same node. In this case this node can be replaced by its success child, node *V*. Figure 4.4 shows an example of the application of this reduction rule. The top node in Figure 4.4a is an example of a BDD that was output from an OR operation between two nodes *F* and *G*:

$$F \vee G = ite(A_2(0,1), U, V),$$

where the reduction rule was not used. The reduction rule can be applied to this BDD since the top node and its success son, node *V*, both have variables from failure mode A_2 and both have node *U* as their failure child. Applying the reduction rule replaces the top node in Figure 4.4a with its success child, node *V*, resulting in the smaller BDD shown in Figure 4.4b.

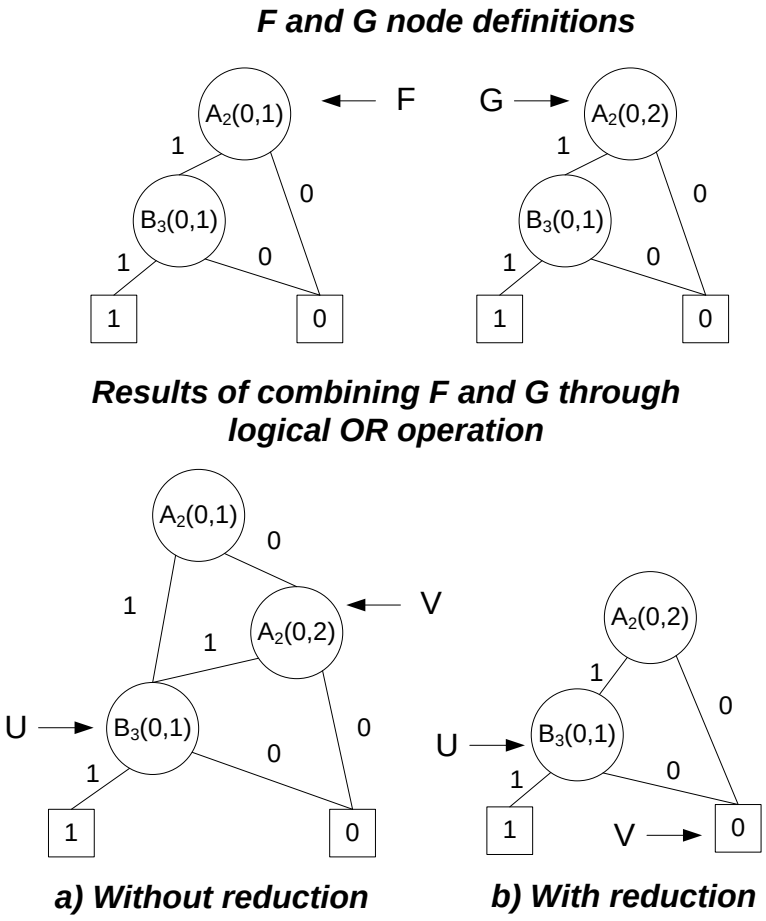


Figure 4.4 – BDD computation result reduces from a) to b) due to reduction rule

The use of this reduction rule means that the number of nodes in the BDD is:

- Often significantly reduced.
- Independent of whether a forward or backward phase variable ordering is used.

4.2.2 BDD Probability Evaluation

If, as is often the case, a BDD path contains multiple events associated with the same component, but from different phases or failure modes, then dependencies will exist between them. In order to evaluate the probability at a BDD node, which is the sum of the probabilities of the paths from that node, these dependencies must be resolved.

In the methods from Zang et al and Tang et al, the probability at a particular BDD node is calculated using a traversal of the BDD. During these traversals, the component, phase and failure mode for the BDD node variable is compared with that of its child node in order to determine where dependencies exist. The corresponding form of the probability calculation is then chosen, based on the dependencies found, and the node probabilities that form part of that calculation are either calculated, in an identical manner, or fetched from the cache if calculated previously. In the method from Zang et al the traversal is limited to comparisons between the variables of the node being calculated and its immediate children, whilst the node probabilities used in the calculations are limited to those from child nodes that are a maximum of two levels below (i.e. the immediate children of the immediate children for the node being calculated). However, in the method from Tang et al the traversals can extend through the complete BDD, terminating at the terminal nodes. Each traversal therefore has potentially high computational expense as many comparison operations may need to be performed in the probability calculation of each node. In addition, due to the inherent coupling between the probability evaluation and dependency resolution, the dependencies must be resolved each time the BDD probability is evaluated (for example, when re-evaluating the BDD due to a change in failure mode failure rates or phase durations). This is costly in terms of computational resource and occurs when calculating importance measures, during real time analysis and when investigating the effect of different component reliabilities for example. The evaluation procedure specified by Tang et al also contains a number of errors that lead to incorrect probability evaluations in certain circumstances. For example, the mission failure probability for the BDD shown in Figure 4.2 is not evaluated as the correct $P(A_1(0,2)) + P(A_2(0,1))$ by their method.

Due to the problems with computational efficiency and accuracy that were identified in the method from Tang et al, a new and improved method was developed. This new method involves the formation of a dependency free data structure, named the Implicant Tree, prior to the probability evaluation step. This data structure is a representation of the mutually exclusive implicants from the BDD, comprising of combinations of independent events, in a form that is primed for fast evaluation. The absence of dependencies during probability evaluation permits higher evaluation efficiency during repeat evaluations such as when calculating importance measures or performing real time analysis.

4.2.2.1 Phase algebra

To construct the Implicant Tree, a set of rules are used to resolve dependencies between events associated with the same component. The four rules were derived by modifying a subset from a larger set of rules originally developed for accident sequence analysis by by Kohda et al [28].

The first two reduction rules, given by Equation 4.2 and Equation 4.3, are used to resolve phase dependencies between events from the same failure mode.

$$P\left(\left(I_j(0,p_1)=0\right)\wedge\left(I_j(0,p_2)=1\right)\right)=P\left(I_j(p_1,p_2)=1\right), \text{ where } p_1<p_2. \quad \textbf{4.2}$$

$$P\left(\left(I_j(0,p_1)=0\right)\wedge\left(I_j(0,p_2)=0\right)\right)=P\left(I_j(0,p_2)=0\right), \text{ where } p_1<p_2. \quad \textbf{4.3}$$

The third and fourth rules, given by Equation 4.4 and Equation 4.5 are used to resolve dependencies between events associated with different failure modes from the same component:

$$P\left(\left(I_{j_1}(0,p_1)=0\right)\wedge\left(I_{j_2}(p_2,p_3)=1\right)\right)=P\left(I_{j_2}(p_2,p_3)=1\right), \text{ where } j_1\neq j_2. \quad \textbf{4.4}$$

$$P\left(\left(I_{j_1}(0,p_1)=0\right)\wedge\left(I_{j_2}(0,p_2)=0\right)\ldots\wedge\left(I_{j_2}(0,p_3)=0\right)\right) \quad \textbf{4.5}$$

$$=1-\left(P\left(I_{j_1}(0,p_1)=1\right)+P\left(I_{j_2}(0,p_2)=1\right)+P\left(I_{j_3}(0,p_3)=1\right)\right),$$

where $j_1\neq j_2\ldots\neq j_n$.

Equation 4.2 expresses that the probability of a component not failing in a certain failure mode before the end of a phase, but failing in that failure mode before the end of some later phase is equal to the probability of failure between the two phases. Equation 4.3 expresses that the probability that a component does not fail in a certain failure mode before the end of two different phases is equal to the probability that it does not fail prior to the end of the later phase. Equation 4.4 expresses that the probability a component does not fail in a failure mode before the end of a certain phase but fails in a different failure mode belonging to the same component at some point between the end of two phases is equal to the probability of this latter event alone. Finally, equation 4.5 expresses that the probability of a component not failing in a set of its failure modes before specific phases is equal to 1 minus the sum of the probabilities of each of the failure modes occurring before the end of the phase that they are successful until.

The only other combinations that can occur on a BDD path are between events belonging to different components and since these are already independent no reduction rules are necessary.

4.2.2.2 Dependency Resolved BDD Structure

To efficiently evaluate the BDD described in the preceding section, it is first converted into another data structure, named the Implicant Tree. The Implicant Tree formed from a BDD node represents its set of mutually exclusive implicants in the form of a set of independent component event combinations.

Overview

The Implicant Tree is a connected acyclic graph with two types of node and a single end (leaf) node. The first type of node, named an Event Node, represents an event, with all Event Nodes except the single end (leaf) node of the tree representing an event for a component. Note that multiple failure mode success events from the same component (see equation 4.5) are considered to be a single component event and are therefore represented by a single Event Node. The tree's end node usually represents the terminal 1 node and is treated as the certain event which has a probability of one. In the special case of the Implicant Tree representing the BDD that consists of a single terminal 0 node, the tree's end (and root) node is a null node that represents the impossible event and has a probability of zero. All event nodes, except the end node, have a single child node but may have multiple parents. An Event Node represents an event from an implicant that belongs to a certain component. One important property of the Implicant Tree is that no path from root to end node passes through an Event Node representing an event from the same certain component more than once; events on a path are therefore fully dependency resolved and independent.

The second type of node is called a Summing Node and, unlike an Event Node, nodes of this type can have multiple child nodes as well as parents. A Summing Node's children all belong to the same component and this component is known as the component of the Summing Node. Summing nodes exist where sets of implicants share common event combinations belonging to certain components.

Implicant Tree Construction

The Implicant Tree for a particular BDD node is formed through the procedure outlined below:

If the node is a terminal 1 node, then its Implicant Tree consists of a single terminal one Event Node representing the certain event. If instead the node is a terminal 0 node then the Implicant Tree is represented by a null node. Otherwise the BDD node is an *ite node* and the Implicant Tree is formed from steps *A*, *B*, *C*, *D* and E below.

Step A – Get the Implicant Trees representing the BDD node's failure and success child. In turn each child node will build its Implicant Tree from the procedure outlined here and this is therefore a recursive process that ends at the terminal nodes. This step is shown in Figure 4.5.

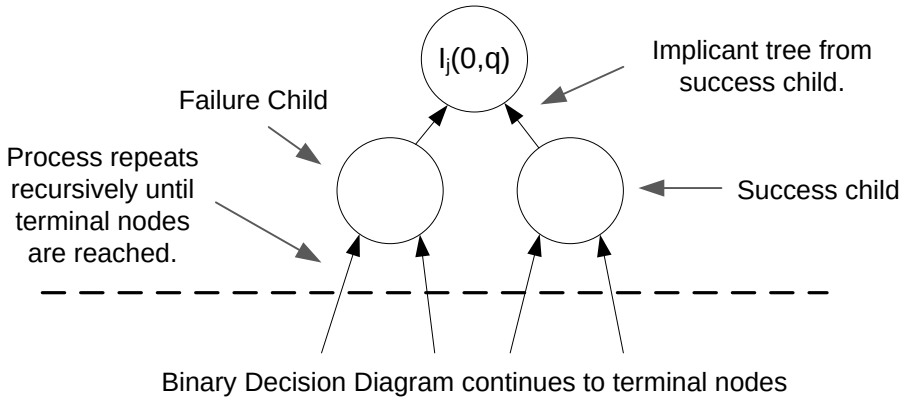


Figure 4.5 – Step *A* of forming an implicant tree for an *ite* node.

Step B – Form the node events that will later be combined (in step C) with the Implicant Trees acquired in step A.

1. The event to combine with the Implicant Tree from the failure (1) child is $I_j(0,q)=1$
2. The event to combine with the Implicant Tree from the success (0) child is $I_j(0,q)=0$.

where *I* is the node variable's component, *j* is its failure mode and *q* is its phase.

An example of this step is shown in Figure 4.6.

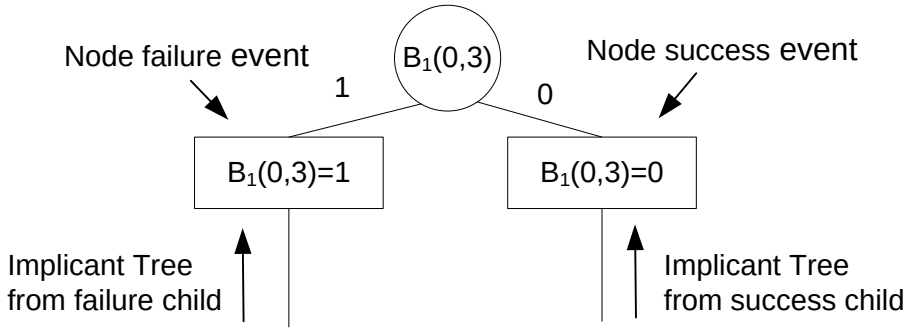


Figure 4.6 – Step *B* of the data structure derivation.

Step C – derive a new set of Implicant Trees by combining the Implicant Trees from step A with the appropriate node event that was generated in step B.

The following steps are carried out for each of the two corresponding implicant Tree and node event pairs:

1. If the Implicant Tree is a terminal one Event Node then create an Implicant Tree consisting of an Event Node representing the node event whose child node is the terminal one Event Node. Alternatively, if the Implicant Tree is a null node then no Implicant Tree is created. For all other Implicant Trees, continue to step 2.
2. If the Implicant Tree and node event belong to the same component, then a dependency exists between the two and rule 3a below should be used to generate the new Implicant Tree. Otherwise use rule 3b (this is always the case with the Implicant Tree from the failure child, due to the BDD construction rules discussed earlier).
3. If the Implicant Tree from step A's root node is a Summing Node then $X_1 \ldots X_n$ designate the Implicant Trees represented by its *n* children (each of which has an Event Node as it's root node). The node event from step B is designated as event *Y*.

a. This step results in *n* Implicant Trees. For *i* = 1 to *n*, find the event representing the intersection of event *Y* and the event represented by the root node of X_i using the rules for combining dependant events that were given in equations 4.2 to 4.5. If combining X_i and *Y* results in the event represented by the root node of X_i then Implicant Tree *i* is X_i. Otherwise, Implicant Tree *i* is formed by creating a new Event Node representing the resultant event as the root node and setting its child node to the child node of X_i.

b. Create a new Implicant Tree with event *Y* as the root Event Node and the Implicant Tree as its child node.

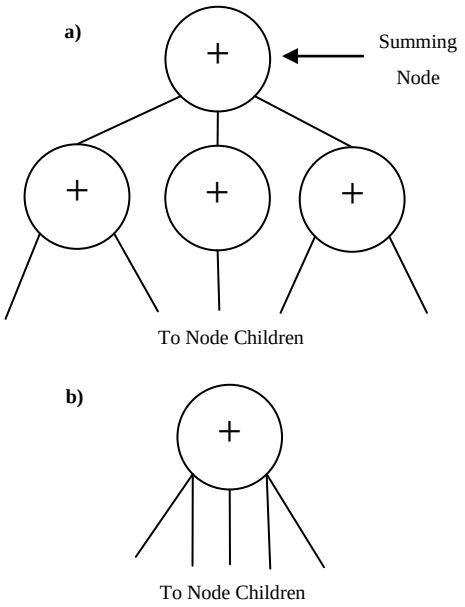


Figure 4.7 – Equivalent Implicant Trees due to merging of summing nodes.

Step D – If only one Implicant Tree was formed in Step C then this step can be skipped. Otherwise, form a Summing Node with each of the Implicant Trees derived in step C as its children, resulting in a single Implicant Tree. Since a Summing Node with a child Summing Node effectively results in a merging of Summing Nodes, this is equivalent to setting the children of the root nodes from the Implicant Trees derived in step C as the new Summing Node's children, as shown by the equivalent Implicant Trees in Figure 4.7a and Figure 4.7b.

Step E – Cache the Implicant Tree derived in step D so that the same Implicant Tree is reused where the BDD node has multiple parents. This is the Implicant Tree representing the BDD node.

The Implicant Tree representing the full BDD is found by forming the Implicant Tree from its top node.

Figure 4.8 shows an example of a BDD, with nodes labelled A – F, and the corresponding Implicant Trees for each BDD node that was formed through the procedure outlined above are shown in Figure 4.9.

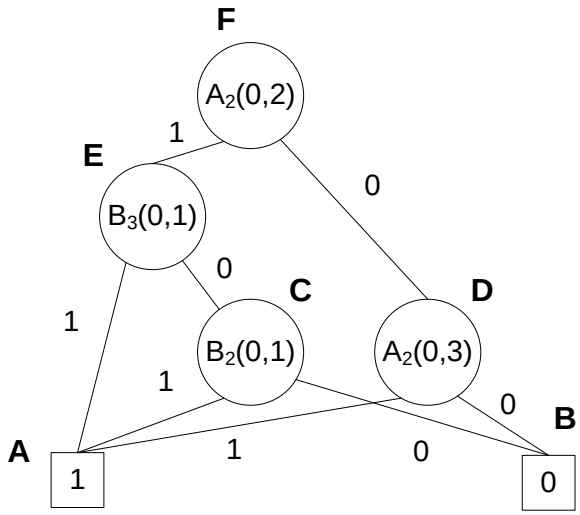


Figure 4.8 – BDD with nodes *A*-*F*.

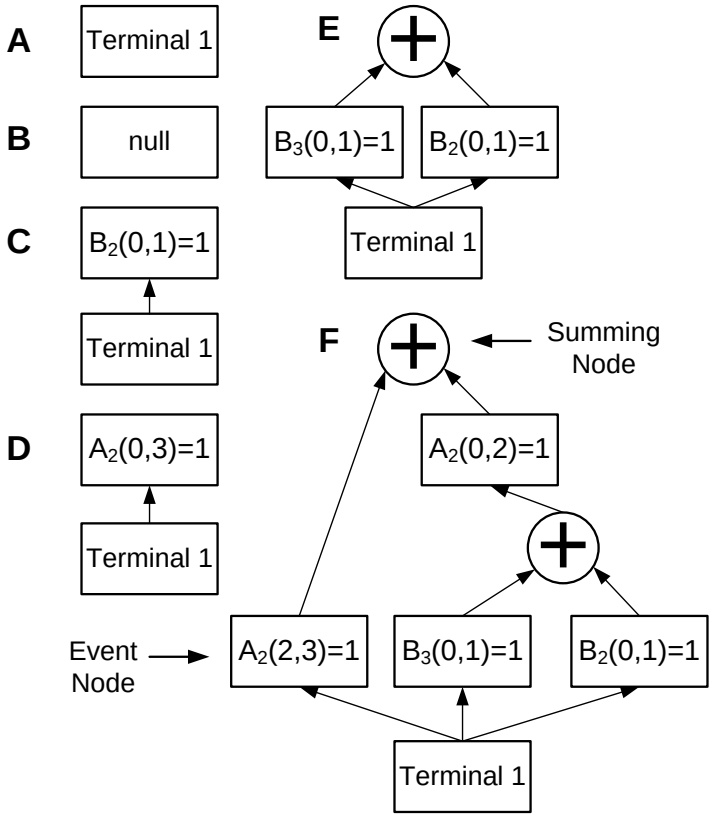


Figure 4.9 – Implicant Trees *A*-*F* corresponding to nodes *A*-*F* from Figure 4.8.

Evaluation of the Implicant Tree

Once the Implicant Tree representing the paths through the BDD has been constructed the evaluation of mission probability is simple and fast. The probability of an event node is calculated as the product of its event probability and its child node's probability, whilst the probability of a summing node is found from the sum of its child node's probabilities. The terminal 1 node is treated as the certain event and therefore has a probability of 1, whilst the null node, which is only ever present in the Implicant Tree representing a single terminal 0 node, has a probability of 0. The probability of each node in the Implicant Tree is cached upon calculation to avoid recalculation when it has multiple parent nodes.

Since the Implicant Tree completely represents the mission reliability structure, it does not need to be rebuilt unless the reliability structure changes. If the Implicant Tree is re-evaluated, for example in real time analysis, the cached values may no longer be correct and must be cleared. The combination of caching and each node's probability being dependent only on its own event and the probability of its immediate children, results in efficient evaluation compared to the phased mission BDD evaluation algorithms which rely on traversing the BDD.

4.2.3 Full Example

The example phased mission shown in Figure 4.10 will be used to demonstrate the method presented in this chapter. This example system has 6 non-repairable components: *A*, *B*, *C*, *D*, *E* and *F*. Each of which has 2 mutually exclusive failure modes, named *1* and *2*, with constant failure rates of 0.001 failures per hour. The duration of each of the three phases is 100 hours.

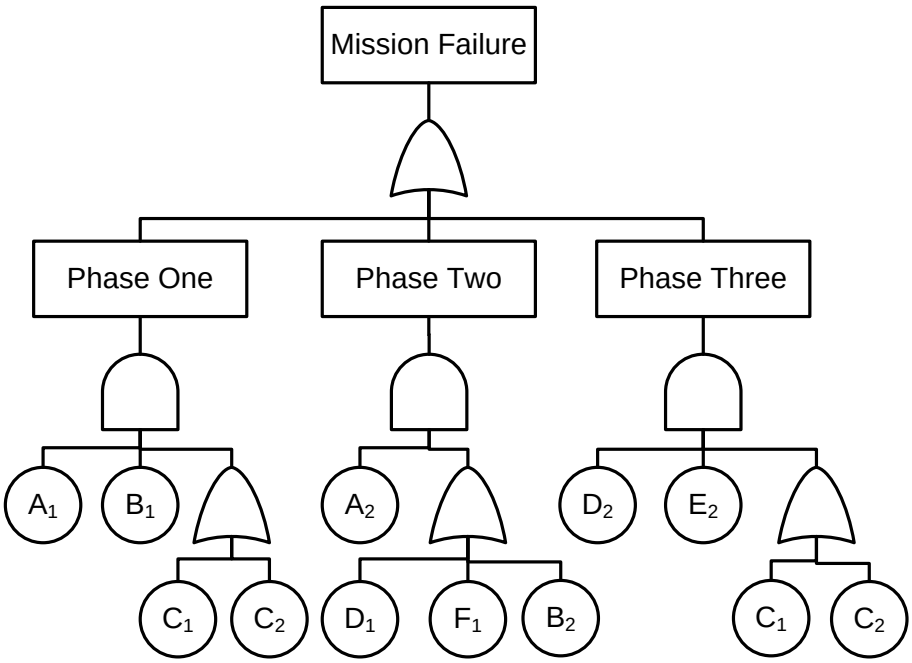


Figure 4.10 - Mission failure fault tree for example phased mission.

The BDD produced from the mission fault tree is shown in Figure 4.11 when constructed using the following, ramdomly chosen, component and failure mode ordering: $B_2<B_1<A_1<A_2<C_2<C_1<D_2<D_1<F_1<E_2$. Note that this BDD does not follow the usual convention of linking nodes to their failure child from the same side (usually left) of the node for all nodes due to the need to produce a clear and concise diagram. The Implicant Tree formed from the BDD is shown in Figure 4.12. The mission unreliability for the example mission, using the component reliabilities given earlier, is 0.116.

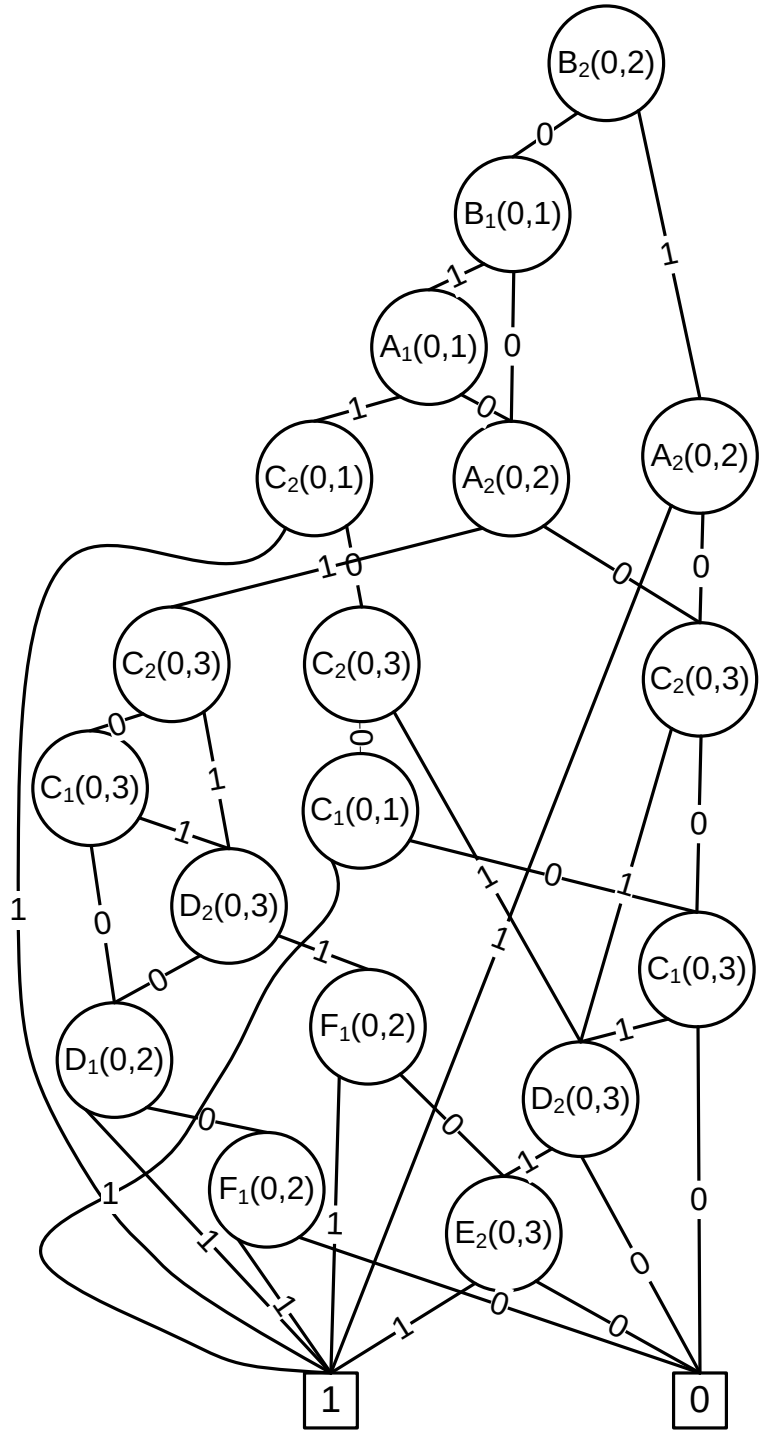


Figure 4.11 – BDD representation of the phased mission for the example phased mission from Figure 4.10.

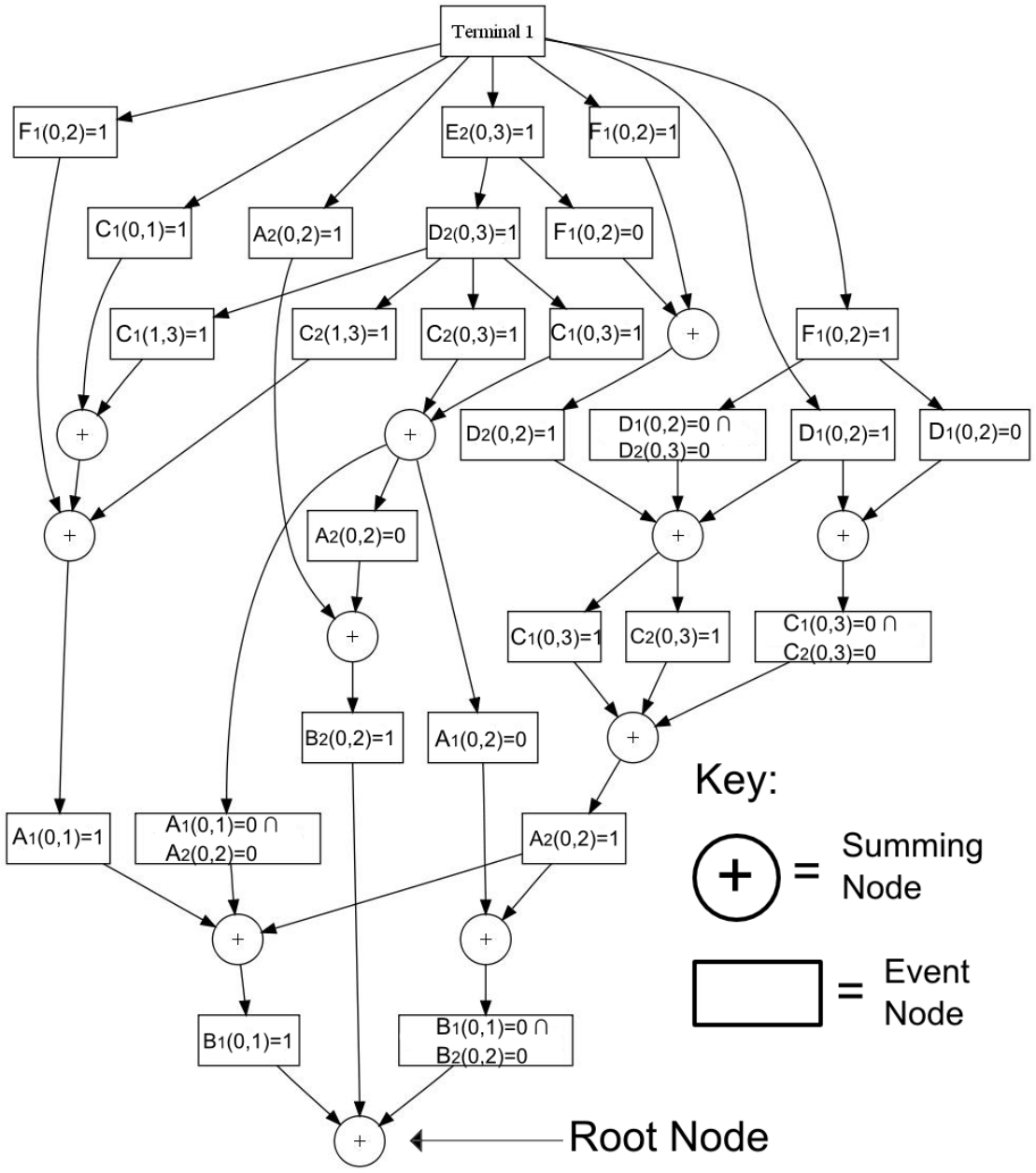


Figure 4.12 – Implicant Tree from from the BDD shown in Figure 4.11.

4.3 Software Implementation

A software tool, implementing the new techniques discussed in the previous section has been developed that can perform many types of complex phased mission analysis. The major part of the software is in the form of a class library, i.e. a collection of related, integrated code files designed to analyse phased missions. It can therefore, potentially, be used in many ways, for example as the analysis engine behind a user interface in a desktop phased mission analysis program or embedded within a hardware system performing reliability monitoring. The software library has been designed to ensure that future developments and improvements can be easily integrated. This has been achieved by constructing it from easily extended, replaceable modules with minimal interdependencies where possible. For example, the ability to read a new fault tree data file format or the

addition of a different BDD computation algorithm require minimal modifications to existing code. Some extensions to its basic function that have already been implemented, such as the calculation of importance measures, are discussed in later chapters of this thesis.

High efficiency in terms of computational resources required, such as processing time and memory requirements, has been an important goal in the development of the software. The reliability analysis of phased missions is a computationally expensive process and an inefficient implementation could limit the applicability of the software to the analysis of only very simple phased missions.

The operation of the software, some implementation details and reasons for the major design decisions will now be explained.

4.3.1 Mission Reliability Analysis Overview

The flowchart shown in Figure 4.13 depicts the basic steps performed by the software when analysing the reliability of a phased mission. The dashed arrows in Figure 4.13 depict referential links, for example the fault tree references the component models since the basic events in the fault tree refer to the component models. Each of the steps shown in Figure 4.13 will be explained in greater detail in subsequent sections.

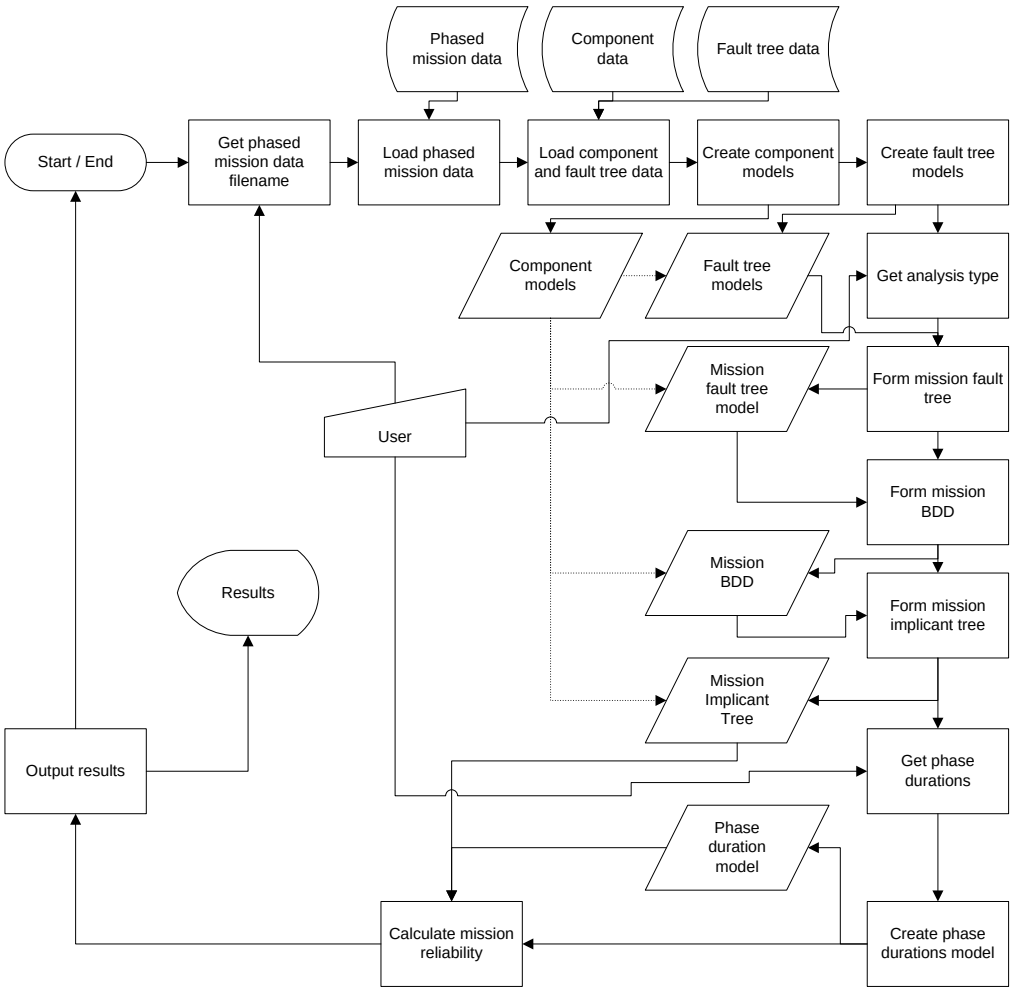


Figure 4.13 – Overview of steps performed to analyse reliability of a phased missions

The user inputs to the program in this example are the phased mission data file, the phase order, the phase durations and the analysis type. The output is the mission reliability in the time period of interest.

4.3.2 File Formats

The software reads two types of data file that describe different data pertaining to the system to be analysed; the first describes a system's components and the second a system's reliability structure in the form of a fault tree. A third type of data file represents a phased mission and contains references to the data files for the missions phase fault trees and the component data file containing the components whose basic events appear in the fault trees.

The component data file contains data on component names, their failure modes and the failure model used for each failure mode. The failure models currently supported are fixed probability (a time independent, hard coded, value), exponential and Weibull. The parameters and their valid value ranges for the supported failure model types are shown in Table 4.2.

Table 4.2 – Parameters for supported component failure models.

Model Type	**Parameter**	**Valid values**
Fixed	Probability	$0 \leq value \leq 1$
Exponential	Failure rate	$value > 0$
Weibull	Shape	$value > 0$
Weibull	Scale	$value > 0$
Weibull	Location	$-\infty \leq value \leq \infty$

Separate files for components and fault trees are used to decouple them and allow a single fault tree to use component data from multiple sources. The phased mission data file does not contain any information regarding the order of the phases as this allows the user to analyse various phase sequences from the same file.

4.3.2.1 XML data files

Although the benefits of standards based file formats for data in any field are well known, no standard file format for the representation of phased missions, fault trees or component reliability models exists. The benefits of adoption of a standard include the ability to share data and the use of independently developed programs to verify the results of analysis.

XML [40] based file formats to represent phased missions, fault trees and component reliability models were developed to be used with the software. File formats based on XML were chosen as they are easily interpreted

by both people and computers, they are easily extensible to include future developments (whilst remaining backward compatible) and parsers for XML are available in the majority of programming languages.

XML Schemas

XML Schemas describe the structure of an XML document, precisely specifying its expected format. The XML Schemas for the fault tree and component data files read by the software are shown in this section.

The XML Schema for a fault tree data file is shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- definition of complex types -->
  <xs:complexType name="basiceventinputtype">
    <xs:sequence>
      <xs:element name="componentName" type="xs:string"/>
      <xs:element name="failureModeName" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="gateeventinputtype">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:group name="eventinputsgroup">
    <xs:sequence>
      <xs:element       name="basicEvent"       type="basiceventinputtype"       minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element        name="gateEvent"        type="gateeventinputtype"        minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>

  <xs:complexType name="inputstype">
    <xs:sequence>
      <xs:group ref="eventinputsgroup" minOccurs="1"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="gatetype">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="type" type="xs:string"/>
      <xs:element name="inputs" type="inputstype"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="faulttreetype">
    <xs:sequence>
      <xs:element name="gate" type="gatetype" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- definition of fault tree -->
  <xs:element name="faultTree" type="faulttreetype"/>

</xs:schema>
```

The XML Schema for component data files is shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- definition of complex types -->
  <xs:complexType name="parametertype">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="value" type="xs:double"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="parameterstype">
    <xs:sequence>
      <xs:element name="parameter" type="parametertype" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="failuremodetype">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="modelType">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Fixed"/>
            <xs:enumeration value="Exponential"/>
            <xs:enumeration value="Weibull"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="parameters" type="parameterstype" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="failuremodestype">
    <xs:sequence>
      <xs:element name="failureMode" type="failuremodetype" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="componenttype">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="failureModes" type="failuremodestype" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="basiceventdatatype">
    <xs:sequence>
      <xs:element name="component" type="componenttype" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

<!-- definitition of basic event data -->
  <xs:element name="basicEventData" type="basiceventdatatype" />

</xs:schema>
```

Shown below is the data file for the phase one fault tree from the example mission that validates against the schema that was previously described:

```
<?xml version="1.0" encoding="utf-8"?>
<faultTree>
  <gate>
    <name>Phase One</name>
    <type>AND</type>
    <inputs>
      <basicEvent>
        <componentName>A</componentName>
        <failureModeName>1</failureModeName>
      </basicEvent>
      <basicEvent>
        <componentName>B</componentName>
        <failureModeName>1</failureModeName>
      </basicEvent>
      <gateEvent>
        <name>G1</name>
      </gateEvent>
    </inputs>
  </gate>
  <gate>
    <name>G1</name>
    <type>OR</type>
    <inputs>
      <basicEvent>
        <componentName>C</componentName>
        <failureModeName>1</failureModeName>
      </basicEvent>
      <basicEvent>
        <componentName>C</componentName>
        <failureModeName>2</failureModeName>
      </basicEvent>
    </inputs>
  </gate>
</faultTree>
```

4.3.3 Software Implementation Details

The software tool has been written in C Sharp (C#) [41] which is a modern object-oriented programming language that is related to C++ [42]. In object-oriented programming, the characteristics of a certain data structure consisting of data fields and methods are defined through a class. The actual realisation of a class, an object whose characteristics are defined through the class definition, is known as an instance. For more information on object orientated program, including advanced concepts such as inheritance and polymorphism used in the implementation, see [43].

4.3.3.1 Components

Components with multiple mutually exclusive failure modes are modelled by the software. In this section the details of the component models are explained.

Component

A Component class instance represents a component in a system. It is modelled as an entity with a name and one or more mutually exclusive failure modes. The component's name is mainly used to identify the component in any output from the program but can also be used by other parts of the software; for example it is used by

certain BDD variable ordering algorithms. There is no limit on the number of failure modes that can belong to each component.

Failure Mode

A FailureMode class instance represents a failure mode of a component. It is modelled as an entity with a name, a component (represented by an instance of the Component class) and a cumulative distribution function (see FailureModel class) representing its time to occurence. Similar to its use in the component model, the name is used mainly for the purposes of program output. The main methods defined for this class are the *MutuallyExclusive* and *FailureProbability* methods. The *MutuallyExclusive* method is given a FailureMode instance as an argument and returns a Boolean result indicating whether the failure mode represented by the argument is mutually exclusive with the failure mode this instance represent, i.e. whether they belong to the same component. The *FailureProbability* method has two arguments that represent the start and end times of a time period, and calls the *Probability* method on its FailureModel instance, returning the result that represents the probability that the failure mode fails during the time period. It also has *GetFailureEvent* and *GetSuccessEvent* methods that have a phase number argument and return a representation of the event that the failure mode occurs or doesn't occur in that phase (see description for FailureModeEvent class).

Failure Model

A FailureModel class instance represents a time to occurence model for a failure mode. The implemented distribution types include the exponential and Weibull distributions. The parameters that must be set for some of the implemented failure model types are shown in Table 4.2. It has a method that returns the probability of failure during a specified time period when given the start and end times of that period, calculated from its parameters and the distribution given by the model type. It can also store and return a reference to another failure model instance and this feature is primarily used to set a temporary failure model for a failure mode - used in more advanced reliability analyses described in later chapters.

It is very simple to expand the software to include a new failure model type, such as the normal distribution, without changing any existing code.

Component Collection

An instance of the ComponentCollection class represents a collection of uniquely named components and is used in the construction of a fault tree from a fault tree data file. The reason for restricting the collection to uniquely named components is that fault tree data files refer to components by name, and therefore ambiguity could arise if multiple components were to share the same name. It has a method that has a filename argument, pertaining to component data file, from which it creates component model instances and adds them to the collection.

4.3.3.2 Fault trees

The software can model fully featured fault trees and the details of these models are explained in this section.

Fault tree nodes

An instance of the FaultTreeNode class is used to represent a gate or basic event node in a fault tree. It implements the properties and methods that are common to all the classes that model fault tree nodes. The most important of these is that both have a method, named *GetBDD*, which returns a representation of a BDD (see description of the BDDNode class in section 4.3.3.4). This method receives an instance of a BDDBuilder class as an argument (see section 4.3.3.4), along with a phase number, which are used to construct its BDD representation. A fault tree itself is represented by the instance of this class that represents the top (or root) node of the fault tree. The subclasses of this class that are used to represent the different types of fault tree gate are described in the remainder of this section.

Gate nodes

The GateNode class is a subclass of the FaultTreeNode class that is used to represent a gate in the fault tree. AND, OR and NOT gates are all currently implemented, as these are the three most commonly encountered fault tree gate types. There are two types of fault tree gate: those that apply a unary operator to a single input and those that apply binary operators to their inputs. The subclasses of this class that model the specific features of those using binary and unary operators will now be described.

Binary Operator Gate Nodes

An instance of the BinaryOperatorGateNode class models gates that apply binary Boolean logic operations between their inputs. They have a reference to an instance of the BooleanBinaryOperator class, explained in section 4.3.3.3, that applies the appropriate Boolean logic. Therefore the only difference between AND and OR gate node models is that they hold references to different subtypes of the BooleanBinaryOperator class. The *GetBDD* method first calls *GetBDD* on each of the input FaultTreeNode instances to get their BDD representations (see BDDNode class described in section 4.3.3.4). The *Computation* method on the BDD Builder instance is then called with arguments of the BDD representations from the first two inputs and the BooleanBinaryOperator class instance. The call to the *Computation* method is then repeated but with the BDDNode instance arguments replaced with the BDD node instances belonging to the next input and the instance returned from the previous call to the *Computation* method, until no inputs remain. The BDDNode instance returned by the last call to the *Computation* method represents the BDD for the gate.

Unary Operator Gate Nodes

An instance of the NOTGateNode class applies a unary Boolean logic operator on its single input and therefore, unlike instances of the BinaryOperatorGateNode class, does not hold a reference to a Boolean Binary Operator model instance. The *GetBDD* method sets a flag indicating that NOT logic is switched on in the *BDDBuilder*, calls the *GetBDD* method on its input node instance to get the BDDNode instance representing the BDD of this NOT gate (equivalent to its input node instance's BDD representation with terminal nodes switched), switches the NOT logic flag back and then returns the BDD node instance. The affect of this flag and reasons for this implementation choice are given in section 4.3.3.4.

Basic event nodes

An instance of the BasicEventNode class is used to represent a basic event in the fault tree. Since a basic event represents the occurrence of a component failure mode, the instance contains a reference to an instance representing the relevant failure mode. The *GetBDD* method calls the *GetFailureModeNode* method on the BDDBuilder instance with arguments of the failure mode instance and phase number to form the BDDNode instance representing the basic event of the failure of the failure mode before the end of the phase, which it then returns.

Phased mission fault tree

A phased mission fault tree, see section 2.8.1, represents the failure of the system within a specific time period of the mission, e.g. a particular phase or the complete mission. Phased mission fault trees therefore have a top gate that applies Boolean logic on its inputs which each represent the occurrence of a fault tree before the end of a phase, as was shown in Figure 2.10. The phased mission fault tree therefore differs from a standard fault tree by including a time aspect through the phase applied to each of the fault trees that represent system reliability structure in that phase. This is represented by 'time' nodes in the fault tree model.

Time fault tree nodes

The TimeTreeNode class models the time based nodes in the fault tree. Two special types of time nodes are used in the construction of phased mission fault trees, represented by the MissionNode and PhaseNode subclasses. Similar to those modelling standard fault tree nodes, this class defines a *GetBDD* method that returns a BDDNode instance representing its equivalent BDD representation. The only difference is that the method has only one argument for time fault tree nodes, a BDDBuilder instance.

Mission nodes

A MissionNode class instance represents a gate in the phased mission fault tree that applies Boolean logical operations on its input nodes to describe the Boolean logic of the tree with respect to times of failure. It is similar to a GateNode, except its inputs are other time fault tree node instances rather than fault tree node instances. The Boolean logical operator applied to its inputs is determined by the instance of the BooleanBinaryOperator class, explained in section 4.3.3.3, that it references. The *GetBDD* method for this node type behaves exactly as it does for GateNode instances, except there is no phase argument in any of the calls to the *GetBDD* method.

Phase nodes

An instance of the PhaseNode class represents a phase fault tree. It has a phase number and a reference to a FaultTreeNode instance, representing the fault tree for the system's reliability structure in that phase. The *GetBDD* method for this node type calls the *GetBDD* method on its input FaultTreeNode instance, giving its phase number as the phase argument. The returned BDDNode instance represents the BDD for the phase fault tree conditions being met before the end of the phase.

Mission Factory

An instance of the MissionFactory class is used to produce MisisonNode instances that represent a mission fault trees for failure of the mission during a particular period, such as the fault tree representing exclusive phase failure, from a set of PhaseNode instances that represent phase fault trees. The types of mission tree representations output from this class include exclusive phase failure, mission success until the end of a phase and mission failure before the end of a phase.

4.3.3.3 Boolean Binary Operators

The models of Boolean binary operators that output a Boolean value from two Boolean input values are described in this section.

Boolean Binary Operators

An instance of the BooleanBinaryOperator class represents a Boolean binary operator. It has a *GetOutput* method that has two BDDNode instances as inputs and returns a BDDNode instance computed from these inputs using the logic of the operator. It also has a *GetReverse* method that returns an instance of the BooleanBinaryOperator subclass whose *GetOutput* method returns the opposite output for opposite inputs.

The ANDOperator and OROperator subclasses have been implemented to represent Boolean AND and OR operations respectively - the two most common types of Boolean binary operator.

AND Operator

ANDOperator class instances represent the AND Boolean binary operator. The output of instances of the *GetOutput* method for the possible combinations of input BDDNode types are shown in Table 4.3. When the operator is applied to two ITENode instances, described in section 4.3.3.4, that represent *if-then-else* (*ite*) BDD nodes, the output cannot be determined by the operator alone.

Table 4.3 – Boolean AND operator truth table for BDDNode instance inputs.

Input Node A Type	Input Node B Type	Output
Terminal One Node	Terminal One Node	Terminal One Node
Terminal One Node	Terminal Zero Node	Terminal Zero Node
Terminal One Node	If-Then-Else Node	Node B
Terminal Zero Node	Terminal One Node	Terminal Zero Node
Terminal Zero Node	Terminal Zero Node	Terminal Zero Node
Terminal Zero Node	If-Then-Else Node	Terminal Zero Node
If-Then-Else Node	Terminal One Node	Node A
If-Then-Else Node	Terminal Zero Node	Terminal Zero Node
If-Then-Else Node	If-Then-Else Node	Requires computation

The *GetReverse* method returns an instance of the OROperator class discussed in the next section.

OR Operator

An instance of the OROperator class represents a OR Boolean binary operator. The outputs from its *GetOutput* method for each input BDDNode type combination are shown in Table 4.4. As with the AND operator model, it cannot determine the output of operations between two ITENode instances.

Table 4.4 – Boolean OR operator truth table for BDDNode instance inputs.

Input Node A Type	Input Node B Type	Output
Terminal One Node	Terminal One Node	Terminal One Node
Terminal One Node	Terminal Zero Node	Terminal One Node
Terminal One Node	If-Then-Else Node	Terminal One Node
Terminal Zero Node	Terminal One Node	Terminal One Node
Terminal Zero Node	Terminal Zero Node	Terminal Zero Node
Terminal Zero Node	If-Then-Else Node	Node B
If-Then-Else Node	Terminal One Node	Terminal One Node
If-Then-Else Node	Terminal Zero Node	Node A
If-Then-Else Node	If-Then-Else Node	Requires computation

The *GetReverse* method returns an instance of the ANDOperator class discussed in the previous section.

4.3.3.4 Binary Decision Diagram

The software has a full implementation of phased mission Binary Decision Diagrams that uses the methods developed in this chapter. The various aspects of the Binary Decision Diagram implementation are described in this section.

BDD Node

The BDDNode class represents a BDD node and defines the methods common to the models for the two specific types of BDD node, the *ite* node and the terminal node, that are modelled by the subclasses described in the remainder of this section. The most important of these methods is the *GetImplicantTree* method that returns a ImplicantTreeNode instance, described in section 4.3.3.6, representing the Implicant Tree for the BDD node.

If-Then-Else Node

An instance of the ITENode class is used to represent an *ite* node in a BDD. It holds a reference to a FailureMode instance and has a phase number which together represent the failure mode and phase of its Boolean variable. It also holds a reference to two BDDNode instances that represent its failure and success child nodes. The *GetImplicantTree* method implementation in this class first gets the ImplicantTreeNode instances representing the Implicant Tree nodes for its failure and success child nodes. Secondly, FailureModeEvent instances representing the failure and success events for the node's Boolean variable are then formed from the FailureMode instance and phase number. In the third step, the *AND* method is called on each

ImplicantTreeNode instance returned from the failure and success child nodes with the FailureModeEvent instances representing the failure and success event used as the argument respectively. If this results in more than one ImplicantTreeNode instance, a SummingNode, described in section 4.3.3.6, is formed with these as inputs and returned from the method. Alternatively, if only a single ImplicantTreeNode instance is obtained then that ImplicantTreeNode instance is returned directly.

Terminal Node

Instances of the TerminalNode class are used to represent the terminal nodes in the BDD. It holds a Boolean value of true or false, the value of which indicates whether the instance represents a terminal one or terminal zero BDD node respectively. The implementation of the *GetImplicantTree* method for this class is very simple; depending on whether the node represents a terminal one or terminal zero BDD node, a TerminalOneNode or TerminalZeroNode representing the Implicant Tree node equivalent is returned.

BDD Builder

An instance of the BDDBuilder class manages the construction of BDDs from fault trees. It is used in the *GetBDD* method of the FaultTreeNode class to create the BDDNode instance that represents the BDD equivalent of the fault tree node.

If-Then-Else Table

ITETable class instances represent an *if-then-else* BDD node table that prevents the creation of identical ITENode instances and ensures that they are instead shared (i.e. given multiple parents) - vital to the efficiency of the BDD method. The decoupling of the BDDBuilder from a single BDD is an important implementation feature that facilitates the sharing of BDD nodes not only within a single BDD but across the BDD representations of multiple fault trees. This can lead to a significant increase in computational efficiency when multiple related fault trees are modelled at the same time. The ITETable class defines two main methods, named *TryFindNode* and *AddLastSearched*. The *TryFindNode* method is given arguments of a FailureMode instance, a phase number and two BDDNode instances that represent failure and success child nodes. Together these arguments define the properties of an *ite* BDD node and the method searches for a matching node in the table, returning true along with the BDDNode instance representing the matching node if found, and false otherwise. The *AddLastSearched* method creates a BDDNode instance with the properties of the *ite* BDD node that was last searched for using the *TryFindNode* method, adds it to the table and then returns it.

Previous Computation Table

An instance of the PreviousComputationClass keeps track of the computations performed during the construction of a BDD, allowing the previously computed result to be used when any computation is repeated. The class defines two important methods, named *TryGetResult* and *AddResultForLastSearched*. The *TryGetResult* method has arguments of a BooleanBinaryOperator instance and two BDDNode instances that define the computation. It searches through the previously stored computations to try and find a match, returning true and the BDDNode representing the computation result if successful and returning false if not. The *AddResultForLastSearched* method is given a BDDNode argument that represents the result of the last

computation that was searched for using the *TryGetResult* method, and then stores it as the result of that computation.

Ordering strategy

To represent an ordering strategy, that determines the relative ordering of *ite* node variables within a BDD, an instance of a subclass of the OrderingStrategy class is used. A method named *GetPriority* is defined within the class that receives two ITENode instances, determines the relative orderings of their variables according to the implemented ordering strategy and returns a variable that indicates which node should appear higher in the BDD.

A subclass named SimpleBDDOrdering that orders variables according to the strategy described in section 4.2.1 has been implemented but it makes no attempt to optimise the orderings of either components or failure modes to minimise the size of the BDD as this is beyond the scope of this work. The process of adding more complex ordering strategies is simple, requiring only the definition of a new OrderingStrategy subclass that defines the *GetPriority* method.

Application of NOT logic

The Boolean logical NOT of a BDD is given by swapping its terminal one and zero nodes, however simply swapping the value of the terminal one and zero nodes in the implementation presents some difficulties and is not the most efficient approach. One problem with this approach is that the results from the table containing previous computations from the BDD construction can no longer be used, reducing the efficiency for future computations. This is because the result stored in the table for a particular computation may now refer to the NOT equivalent of the true result. A second problem is that any other BDDs produced by the same BDDBuilder instance would also unintentionally become their NOT logic equivalents, as they will share the same terminal nodes. An alternative approach, avoiding these problems, would be to create a copy of the BDD but with the terminal nodes swapped. This approach is also inefficient due to the need to produce two BDDs and the lack of a previous computation table produced for the NOT logic BDD resulting in lower efficiency for future calculations. In the most efficient approach, the BDD with swapped terminal nodes is built directly, without the prior creation of its equivalent that is necessary in the alternatives. This is achieved in the implementation through the setting a flag in the BDDBuilder instance that indicates that NOT logic is on, causing it to produce the NOT equivalent BDD representation of a fault tree node until it is switched off. The switching of this flag was discussed in section 4.3.3.2.

Creation of BDD node from Fault Tree Node

The main role of a BDDBuilder instance is to create BDDNode instances from FaultTreeNode instances, i.e. converting fault tree representations into their BDD equivalents. The implementation of the conversion process for BasicEventNode and BinaryOperatorGateNode class instances is given in this section.

Basic Event Fault Tree Node

As was discussed in section 4.3.3.2, BasicEventNode instances call the *GetFailureModeNode* method on a BDDBuilder instance, passing its FailureMode instance and phase number as arguments, which then returns the

BDDNode instance representing its BDD. The *GetFailureModeNode* method forms a ITENode instance with the FailureMode instance and phase number, defining the BDD node variable, and failure child and success child nodes consisting of the terminal one and terminal zero nodes respectively. If the NOT logic flag is on then the orientation of the terminal one and terminal zero nodes is the reverse.

Gate Fault Tree Node - BDD computation algorithm

BinaryOperatorGateNode instances call the *Computation* method on a BDDBuilder instance with arguments of its BinaryOperator instance and the BDDNode instances representing its input fault tree nodes, as was discussed in section 4.3.3.2. The first step in the *Computation* algorithm is performed only if the NOT logic flag is on. In this step the *GetOpposite* method on the BinaryOperator instance is called to get its logical opposite, the flag is then switched off, and the *Computation* method called again this time using the new BinaryOperator instance with the other arguments the same, before switching the flag back to on. This has the effect of swapping all AND gates for OR gates, and vice versa, and all failure basic events into their success event equivalents in the fault tree from which the BDD node representation is calculated – thus implementing the NOT logic. The second step is to check whether the computation has been previously performed, using the previous computation table discussed in section 4.3.3.4. The previously calculated result is returned directly if a matching computation is found, otherwise it must be calculated. The first step in the calculation process determines whether the result can be determined from the *GetOutput* method of the BinaryOperator instance with arguments of the two BDDNode instances. If a result is determined then it is returned, otherwise further calculation is required. In this case an algorithm builds the BDDNode instance representing the resultant BDD according to the method developed in this chapter and returns the result. This algorithm includes the steps for minimising the BDD size according to the phase and failure mode algebra, along with the developed reduction rules.

Phase durations

An instance of the MissionTimes class represents a set of phase durations for a phased mission. A number of methods are defined for this class that allow them to be modified and values, such as a phase end time, to be obtained.

4.3.3.5 Component Event

The software models the various types of events that can occur for a multiple failure mode component during a phased mission. These are described in this section.

Component Event

Subclasses of the ComponentEvent class represent failure and success events, pertaining to a single component, during a particular period of the mission that is defined by the phase ends it occurs after and before. This class defines the methods and properties that are common to all of these subclasses. The most important of these are the *Probability* and *SameComponentAND* methods. The *Probability* method is given a MissionTimes instance as an argument and, and using the phase durations that it represents, returns the probability of the event occurring during the mission. By using an argument to this method that represents the phase durations, and defining the period in which the event occurs in terms of the phase ends rather than actual times, the event's probability for any set of phase durations can be evaluated by using the appropriate MissionTimes instance. A

NOTProbability method is also defined that returns the probability that the event doesn't occur, calculated from unity minus the output of the *Probability* method. The *SameComponentAND* method is given an argument of a FailureModeEvent instance, representing an event from a single failure mode and discussed in the following section, that must belong to the same component as this event and returns a ComponentEvent instance that represents the result of a Boolean logical AND operation between the two. The subclasses of this class that represent events for a single failure mode and success events for multiple failure modes belonging to the same component are discussed in the remainder of this section.

Failure Mode Event

Subclasses of the FailureModeEvent class represent events for the failure or success of a single failure mode of a component. They have a reference to a FailureMode instance, representing the failure mode of the event, and methods named *FailsAfter* and *FailsBefore* that return numbers indicating the phase ends that the failure mode occurs after and before respectively, according to the event. The *SameComponentAND* method is defined in this class for the case that the argument event belongs to the same failure mode as the event instance on which it is called (i.e. where the phase algebra rules from Equations 4.2 and 4.3 apply), whilst the other cases (i.e. the application of the phase algebra from Equations 4.4 and 4.5) are dealt with in the subclasses that are described below. In all cases the output of the method follows those phase algebra rules.

Failure Mode Occurrence Event

An instance of the FailureModeOccurrenceEvent class represents the event that a failure mode occurs in a certain phase period. It has two phase numbers representing the phase end that the failure mode occurs after and before. The *Probability* method first converts the phase numbers into phase end times using the *GetPhaseEndTime* method of the MissionTimes instance and then calls the *FailureProbability* method of the FailureMode instance with these time arguments to determine the probability of failure, which it then returns.

Failure Mode Success Event

The event that a failure mode does not occur during a phase period is represented by an instance of the FailureModeSuccessEvent class. Since it is assumed that components are non-repairable, the event must represent success from the start of the mission, and therefore it has only the phase number representing the phase end that the failure mode does not occur before. Since the event that a failure mode does not occur during a certain period indicates it does occur sometime after that period, the *FailsAfter* and *FailsBefore* methods return the phase end that the failure mode does not occur before and infinity respectively. The *Probability* method first converts the phase number into a phase end time and then calls the *SuccessProbability* method of the FailureMode instance with this time argument to determine the probability of success, which it then returns.

Multiple Failure Mode Success Event

An instance of the MultipleFMSuccessEvent class represents the event that more than failure modes do not occur in defined phase periods. It stores references to a FailureModeSuccessEvent instance and another MultipleFMSuccessEvent instance that together represent the set of failure modes that do not occur. The *Probability* method first calls the *NOTProbability* methods on the FailureModeSuccessEvent and MultipleFMSuccessEvent with the MissionTimes instance argument. The results from these method calls are

summed to give the probability that any of the failure modes occur during their respective time periods and this is then deducted from unity to give the probability that they all do not occur, which is then returned.

4.3.3.6 Directed Acyclic Implicant Graph

The modelling of the Implicant Tree data structure is described in this section.

Implicant Tree

Implicant Trees are represented through instances of subclasses of the ImplicantTreeNode class. Two main methods, named *Probability* and *AND*, are defined for this class. The *Probability* method is given a MissionTimes instance argument, representing the phase durations of the mission, and uses this to calculate, cache and return the probability of the Implicant Tree. The *AND* method is used to perform *Step C* of BDD node to Implicant Tree node construction that was explained in section 4.2.2.2. It is given a FailureModeEvent argument, representing a failure mode event, and returns a ImplicantTreeNode instance representing the Implicant Tree node output from the combination of the event and the Implicant Tree node represented by the instance on which it is called. A method is also defined for clearing the cached probability values in the Implicant Tree nodes so that probability re-evaluation can be performed. The subclasses that represent Event Nodes, Summing Nodes, Terminal One Nodes in an Implicant Tree are described in the remainder of this section.

Event Node

Instances of the EventNode class are used to represent Event Nodes in an Implicant Tree. They store a reference to a ComponentEvent instance that represents its component event and a ImplicantTreeNode instance that represents its child node. The *Probability* method calculates the nodes probability by calling the *Probability* methods on its ComponentEvent and ImplicantTreeNode instances and multiplying the results returned. If the *AND* method is called on an instance of this class with an argument event that belongs to the same component, then the *SameComponentAND* method is called on the argument event with the event from the instance as the argument, and a new EventNode instance returned that references the returned event as its event and has a child node matching the child node of this instance. Alternatively, if they belong to different components, then a new EventNode instance is returned that references the argument event as its event and has the instance as its child node.

Summing Node

Instances of the SummingNode class are used to represent Summing Nodes in a Implicant Tree. They store references to the ImplicantTreeNode instances that represent their children. The probability of the node is calculated in the *Probability* method by calling the *Probability* method on the ImplicantTreeNode instances that represent its children, passing the MissionTimes instance as the arguments, and then summing the returned results. When the *AND* method is called on an instance of this class with an argument event that belongs to the same component, then the *AND* method is called on the ImplicantTreeNode instances that represent its children, passing the FailureModeEvent instance as the argument, and a new SummingNode instance returned that references the returned ImplicantTreeNode instances as its child nodes. If they instead belong to different

components, then a new EventNode instance is returned that references the argument event as its event and has the instance as its child node.

Terminal One Node

An instance of the TerminalOneNode class represents a terminal one node in an Implicant Tree. The *Probability* method always returns a value of 1, since this node type represents the certain event. Calling the *AND* method on an instance of this type returns a new EventNode that references the argument event as its event and the instance as its child.

4.3.3.7 Mission Failure Probability Evaluation

As discussed earlier in this chapter, analysing the probability of failure of a phased mission in a particular period is a multistep process that involves the creation of the appropriate mission fault tree, conversion into a BDD and the conversion into an Implicant Tree. The classes described in the remainder of this section encapsulate some of the steps of this process and facilitate the analysis of the mission failure in precise periods of the phased mission.

Top Event Probability

An instance of the TopEventProbability class is the most general of the classes described in this section and can be used to automate the process of deriving the top event probability from a phased mission fault tree. It holds references to the MissionNode instance that represents the phased mission tree that is to be analysed and a BDDBuilder instance that is used to construct the BDD during the probability analysis. It defines a *Probability* method that has a MissionTimes instance argument, representing the phase durations of the mission, and is used to determine the top event probability for that tree according to those phase durations. Two subclasses of this class are defined for analysing the probability of mission failure in time periods that cannot be represented by a phased mission tree alone, namely the probability of failure on transition to a phase and the probability of in-phase phase failure, and will now be described.

Transitional Probability

The TransitionalProbability class allows the probability of failure on transition to a specific phase to be determined from the set of phase fault trees and phase durations for a mission. An instance of this class holds a reference to a phase number that indicates the phase of the mission to be analysed and is given a MissionFactory instance, representing a set of phase fault trees for a mission, from which it forms and stores a reference to a MissionNode instance that represents the exclusive failure of the mission in that phase. The *Probability* method then calculates the probability of failure on transition to the phase of the mission by first creating a copy of the argument MissionTime instance but with the duration for that phase set to 0, and then uses that with the *Probability* method of the MissionNode instance to get the probability of exclusive transitional phase failure.

In-Phase Probability

The InPhaseProbability class is used to find the probability of in-phase failure in a specific phase from the set of phase fault trees and phase durations for a mission. It is constructed with a MissionFactory instance and phase number from which it forms and stores a reference to a TransitionalProbability instance for finding the

probability of failure on transition to that phase. It also forms and stores a reference to the MissionNode instance that represents the exclusive failure of the mission in that phase. The *Probability* method then calculates the probability of in phase failure in the phase from the difference between the overall probability of exclusive phase failure, calculated from the MissionNode instance, and the probability of failure on transition to the phase, calculated from the TransitionalProbability instance. This is an example of where the probability from the same Implicant Tree is re-evaluated, as both the overall and transitional failure probability calculations are performed on the same Implicant Tree with only the duration of the analysed phase changing. Since the re-evaluation of a Implicant Tree is very fast, there is only a small proportional increase in the amount of time taken to analyse in-phase probability in a phase compared to the overall probability of exclusive phase failure.

4.3.3.8 Phased Mission

The PhasedMission class provides an easy to use interface for the analysis of a phased mission. It provides methods for loading component data, loading phase fault tree data and analysing mission failure during any period of a phased mission. A user interface built on top of the class library would therefore only need to interact with the simple methods of this class and display its outputs, insulating it from the internal details and steps taking during the analysis of a phased mission.

4.3.3.9 Verification and Unit Testing

The correct operation of the methods developed and their implementation has been verified through the use of a large number of unit tests. Each of these verifies the correct operation of a specific part of the programme by comparing its output for various inputs against their known correct results. Due to the successful passing of all tests, the accuracy of the developed methods and their implementation is assured.

4.4 Comparison to method by Tang and Dugan

In order to compare the performance of the method introduced here with the method by Tang and Dugan, referred to as the TD method, it is necessary to test them on a large number of different fault trees. A program was written to randomly generate phased missions with certain properties. The properties that could be varied include the number of components and failure modes available, the depth of the phase fault trees, the proportion of AND to OR gates, the maximum number of gate and basic event inputs to each gate and the number of phases in the mission. The missions were analysed for both overall mission failure probability and exclusive phase failure in the final phase (i.e. system survives until final phase and fails in that phase) as the solution of the latter often has higher computational expense.

The accuracy of the method presented in this paper has been verified against the results published in Zang et al [34] and Tang and Dugan [36]. The TD method gave an incorrect result for some of the additional test cases including the example phased mission shown in Figure 4.10.

A comparison has been made of both the performance and the accuracy. The three metrics used in the comparisons are:

1. Probability value (to compare accuracy)
2. Number of nodes in BDD (to compare BDD construction efficiency)
3. Solution time (to compare evaluation efficiency)

Table 4.5 shows the settings for each configuration that were used to generate the 10 random missions. This set of mission configuration settings were chosen to provide a range of component sets, phase counts, AND/OR gate ratios and phase fault tree shapes (i.e. variations in depth and breadth). Generally, the complexity of the mission increases with fewer components, higher numbers of failure modes, larger phase fault trees and higher numbers of phases. Fewer components and larger numbers of failure modes cause an increase in the number of failure mode dependencies between basic events in the mission fault tree, whilst higher numbers of phases increase phase dependencies. Larger numbers of gates and greater tree depth increase the size of the problem, causing a general increase in complexity. For these reasons, configuration settings *I* and *F* can be considered to generate the most and least complex phased missions respectively.

Table 4.5 – The settings used in the random generation of 10 phased missions.

Configuration	Component count	Failure mode count	Tree depth	Gate inputs	Basic event inputs	OR/AND gate ratio	Phases
A	10	5	5	3	2	0.5	4
B	10	5	5	3	2	0.5	12
C	20	5	5	3	2	0.5	16
D	20	5	5	3	2	0.9	16
E	20	5	5	3	2	0.1	16
F	20	3	5	3	2	0.5	4
G	20	5	5	5	2	0.5	16
H	20	3	3	3	3	0.5	16
I	20	8	8	3	2	0.5	16

The average solution times for each of the methods, given in elapsed clock ticks, are shown in Table 4.6. As shown the new method gives a significant performance boost for all except configuration settings *F*, the simplest

case, where it shows reduced performance. For example, with configuration settings *G* the evaluation time is reduced by 98% on average compared to the TD method.

Table 4.6 - Average solution time comparison between the new and TD methods for each of the configuration settings.

	Mission Failure		**Final Phase Failure**		
Configuration	**New method**	**TD method**	**New method**	**TD method**	**Overall improvement (% reduction)**
A	1320	6691	1187	1462	69
B	3433	12672	2727	12474	76
C	6852	77085	3280	72489	93
D	1535	1925	653	1705	39
E	708	10980	570	18890	95
F	6473	1979	5863	2916	-152
G	3082	82168	2330	149541	98
H	1200	3477	692	4351	76
I	16171	312354	14403	464165	96

Table 4.7 shows the average BDD node count for each of the methods with each mission configuration type. For every mission configuration tested, the new method resulted in a smaller BDD, often significantly such as in configurations *G* and *H* where the node count was reduced by 97% on average. As with the solution times, the magnitude of the improvement is greatest in the configurations with higher levels of basic event dependencies, as shown by the relatively small 4% size reduction achieved in the simplest configuration.

Table 4.7 - Average BDD node count comparison between the new and TD methods for each of the configuration settings.

	Mission Failure		Final Phase Failure		
Configuration	**New method**	**TD method**	**New method**	**TD method**	**Overall improvement (% reduction)**
A	78	101	130	144	15
B	175	1102	293	1045	78
C	264	5936	386	5727	94
D	89	182	92	167	48
E	57	793	118	1374	92
F	200	214	304	311	4
G	171	5950	314	10859	97
H	82	320	124	411	72
I	492	21658	956	33719	97

Finally, Table 4.8 shows the average percentage error in the probability value given by the TD method for each of the phased mission configuration settings. This shows that the accuracy of the TD method is very poor, particularly when complex phased missions are analysed, and this is due to the errors in that method that have been identified.

Table 4.8 - Average percentage error in the solution given by the TD method for each of the mission configuration settings.

Configuration	Mission Failure Probability Error (%)	Final Phase Failure Probability Error (%)
A	9	23
B	22	24
C	32	111
D	10	22
E	73	107
F	2	6
G	65	117
H	22	74
I	308	44

4.5 Summary and Conclusions

The literature review revealed a number of methods that had been developed for the analysis of non-repairable phased mission reliability. The most recent was developed by Tang and Dugan [36] and designed for the analysis of systems containing components with multiple failure modes. However, analysis of the method found that it had a number of problems that resulted in it often giving very poor accuracy in its assessment of the reliability of a phased mission. The need to perform real time or importance measure analysis, that each require the repeated reliability analysis of the same system, was noted as an important application for phased mission reliability analysis methods. For these reasons, research into a method capable of accurate and highly efficient analysis of a phased mission system containing multiple failure mode components was carried out. This resulted in the development of a new method that is particularly efficient when performing repeat analysis and improves on the method by Tang et al in all areas. The BDD construction algorithm results in smaller BDDS reducing the computational memory it requires to store, uses a very compact phase algebra to produce a new pre-evaluation and dependency resolved structure, named an Implicant Tree, and features an evaluation method that is highly optimised. By resolving all dependencies prior to evaluation, instead of resolving dependencies during evaluation as is the case with the method by Tang et al, repeat evaluations, such as those performed in importance measure analysis, are more efficient. The method has been implemented as a computer code and its performance compared to an implementation of the method developed by Tang et al by applying them to the analysis of a wide range of example phased mission systems. The new method was shown to be an improvement in terms of both efficiency and accuracy.

5 Literature review of reliability methods for repairable phased missions

Certain systems operating in phased missions have components that can be repaired during the mission to increase its reliability. This chapter is a review of the available methods for the analysis of the reliability of such systems. These methods use the Markov, binary decision diagram (BDDs) and Monte Carlo simulation techniques.

5.1 Markov model based methods

The Markov modelling technique has been widely used to analyse the reliability of repairable phased missions. Phased mission Markov models must be able to model transitions between phases and minimise the state space to avoid the computational expense that can occur when modelling large systems and many phases. Various methods have been developed and they can be split into two strategies; those that use a separate state transition matrix for each phase of the mission and those that use a single state transition matrix for the entire mission. Each of these approaches has their own distinct advantages and disadvantages. Common to these methods is the assumption that failed components are integrated back into the system immediately upon repair.

5.1.1 Methods with a separate transition matrix for each phase

In this section methods that use a separate transition matrix for each phase are discussed. This is a natural approach since the states which represent failure and the transition rates between them vary between phases. The use of separate phase transition matrices reduces the maximum matrix size compared to the use of a single matrix for the complete phased mission. This along with the minimisation of the individual phase matrices reduces the computational expense.

The earliest Markov modelling method for analysing repairable phased missions was developed by Alam and Al-Saggaf [44]. In that method the Markov modelling technique is used to determine the state of the system at the end of a phase, and correspondingly the initial conditions for the following phase, from the initial conditions and the transition rates between states. Repeating this process for each phase of the mission allows the probability of failure in each phase and the probability of the system residing in each state at the end of the mission to be found. Additionally, the probability of the system failing due to each separate system failure mode in each phase can be found.

The first step in implementing the method is to determine the exhaustive set of states in which the system can reside. Since in each state a component is either working or failed (the possibility of multiple failure modes is not considered), the total number of states is 2^{N_C}, where N_C is the total number of components in the system. A 2^{N_C} by 2^{N_C} transition-rate matrix, β, is then formed containing the transition rates between these system states. The element at row *i* and column *j* of β, $\beta(i, j)$, for $i = 1 \ldots 2^{N_C}$ and $j = 1 \ldots 2^{N_C}$ and $i \neq j$ is set to the rate of transition from state *j* to state *i*, $\lambda_{j,i}$, as shown in Equation 5.1.

$\beta(i, j) = \lambda_{j,i}$, where $i \neq j$. **5.1**

Each element on the diagonal of β , $\beta(i,i)$, for $i = 1 \ldots 2^{N_C}$, is set to the negation of the sum of the other elements on column *i* and represents the transition rate out of state *i*, as shown in Equation 5.2.

$$\beta(i,i) = -\sum_{j=1, j\neq i}^{2^{N_C}} \beta(j,i) \quad \textbf{5.2}$$

From this overall transition-rate matrix, β , which includes all possible transitions between states, transition-rate matrices for each individual phase of the mission are formed by setting to zero all repair rate transitions out of states that represent system failed states in that phase. Therefore the transition-rate matrix for phase *k*, β_k , is as shown in Equation 5.3.where $\phi_{k,j}$ is a Boolean variable for state *j* during phase *k* with a value of 1 if it is a system failure state and a value of 0 if it is a system success state.

$$\beta_k(i, j) = \begin{cases} \beta(i, j) & \text{if} \quad \phi_{k,j} = 0 \\ 0 & \text{otherwise} \end{cases} \quad \textbf{5.3}$$

Preventing the system from leaving the states representing system failure is required since a phased mission ends as soon as the system enters a failed state. Such states are known as 'absorbing'. Any of the standard structural reliability methods may be used to ascertain which states represent system failed states, such as the fault tree or reliability block diagram methods.

A length 2^{N_C} column vector for phase *k*, $P_k(t)$, where *ith* element represents the probability that the system resides in state *i* at time *t* from the start of the phase, is used to represent the status of the system and is referred to as the state probability vector. The initial state probability vector, $P_1(0)$, representing the state of the system at the start of the mission, contains the probability that the system starts the mission in each state. Usually the element corresponding to all components working is given a value of 1, whilst all others are given a value of 0, based on the assumption that all component are working at the start of the mission. The state probability vector for the end of phase *k* is then given by Equation 5.4, using the duration of phase *k*, $t_k - t_{k-1}$, the initial probability vector for phase *k*, $P_k(0)$, and the phase *k* transition-rate matrix, β_k . Equation 5.4 assumes constant failure and repair rates are used for all components.

$$P_k(t_k - t_{k-1}) = e^{\beta_k(t_k - t_{k-1})} P_k(0) \quad \textbf{5.4}$$

Once the state probability vector at the end of phase *k*, $P_k(t_k - t_{k-1})$, has been found, the probability of phase *k* in-phase system failure, Q_k^P, is obtained by summing the elements that correspond to system failed states in that phase. This is shown in Equation 5.5 where $P_{k,i}(t_k - t_{k-1})$ is the *ith* element of $P_k(t_k - t_{k-1})$.

$$Q_k^P = \sum_{i \in B} P_{k,i}(t_k - t_{k-1}), B = \{j : \phi_{k,j} = 1\} \quad \textbf{5.5}$$

The probability of failure on transition to the next phase, phase *k+1*, is denoted by Q_{k+1}^T, and calculated as the sum of the elements from the state probability vector at the end of phase *k*, $P_k(t_k - t_{k-1})$, that correspond to working states in phase *k* and failed states in phase *k*+1. This is shown by Equation 5.6.

$$Q_{k+1}^T = \sum_{i \in B} P_{k,i}(t_k - t_{k-1}), B = \{j : \phi_{k,j} = 0 \; and \; \phi_{k+1,j} = 1\} \quad \textbf{5.6}$$

In order for the system to progress from phase *k* to phase *k+1* it must be in a system success state for both phases. The probability that the system is in state *i* at the start of phase *k* + 1, $P_{k+1,i}(0)$, is therefore found from probability that the system is in state *i* at the end of phase *k*, $P_{k,i}(t_k - t_{k-1})$, through Equation 5.7.

$$P_{k+1,i}(0) = \begin{cases} P_{k,i}(t_k - t_{k-1}) & if \; \phi_{k,i} + \phi_{k+1,i} = 0, \\ 0 & otherwise. \end{cases} \quad \textbf{5.7}$$

The end of phase probability vector for the next phase is then found through the same process, i.e. using the state probability vector, phase transition-rate matrix and phase duration in Equation 5.4. This process is repeated for each phase of the mission until the state probability vector corresponding to the mission end time has been determined. The mission unreliability, Q_{Miss}, is then calculated as 1 minus the sum of the probabilities that the system is in a working state at the end of phase *N*, where *N* is the number of phases in the mission, as shown in Equation 5.8.

$$Q_{Miss} = 1 - \sum_{i \in B} P_{N,i}(t_N - t_{N-1}), B = \{j : \phi_{N,j} = 1\} \quad \textbf{5.8}$$

For large systems, this method can suffer from the 'state explosion' problem, as every possible system state is included in each phase's transition-rate matrix. Kim and Park [45] proposed a Markov model method, building upon the work of Alam and Al-Saggaf, that produces a smaller state space compared to the earlier methods by not including the absorbing, system failed, states in the phase transition-rate matrices. The removal of the absorbing states reduces the memory and computational overhead compared to Alam and Al-Sagaf's method, resulting in faster computation of mission unreliability with no loss in accuracy. A disadvantage with this approach is that, unlike the method from Alam and Al-Sagaf, it is not possible to determine the probability of the system failing in a particular failed state, only the overall probability of phase failure can be determined.

However, for most types of analysis, only the overall probability of failure during the mission or in a particular phase is of interest and so the use of this compact state model with its computational benefits is often advantageous.

The first steps in Kim and Park's method are the same as the earlier method from Alam and Al-Saggaf , i.e. determine all possible system states and form the full 2^{N_C} by 2^{N_C} transition-rate matrix, β. A reduced transition-rate matrix is then formed for each phase of the mission, consisting of only the rows and columns from β that represent transitions from and to states that are working states in that phase. The columns containing transition rates from failed states can be removed since the mission ends as soon as the system fails, and therefore a transition out of a failed state is impossible. The rows containing transition rates to failed states can be removed because the rates are contained within the diagonal elements, representing the transition rates out of the working states, of the reduced matrix. The reduced transition-rate matrix for phase *k* will therefore be of order N_{WS_k} by N_{WS_k}, where N_{WS_k} is the number of working states in phase *k*.

The remaining steps are similar to Alam and Al-Saggaf but with state probability vectors that vary in size and whose corresponding states must be determined between phases. A length N_{WS_k}column vector for phase *k*, $P_k(t)$, is used to represent the probability that the system resides in each of the working states from that phase at time *t* from the start of the phase. Since there is no longer a one-to-one correspondence between elements in the reduced state transition-rate matrices from different phases, there is also no longer a one-to-one correspondence between the elements in their state probability vectors. An indexing method is therefore required so that the elements corresponding to the same system state in the reduced state probability vectors from different phases can be mapped to one another. The working states in each phase are re-ordered sequentially with the first working state numbered 1, as shown by Equation 5.9, where $\rho_k(i)$ is the re-ordered index of state *i*, *A* is the set of original indexes of working states from phase *k* and N_{WS} is the total number of working states in phase *k*.

$$1 \leq \rho_k(i) \leq N_{WS_k}, \qquad i \in A \tag{5.9}$$

The original state index (the index that includes all states, both working and failed) for each re-ordered working state index must be retained so that the working state indexes can be mapped back to the corresponding full state index. In other words, there must be a one-to-one mapping between the original and re-ordered indexes of working states in each phase. This is shown by Equation 5.10, where $\delta_k(i)$ is the original phase *k* state index of re-ordered index *i*, N_{WS} is the total number of working states in phase *k* and *A* is the set of original indexes of working states from phase *k*.

$$\left\{\delta_k(i), \qquad 1 \leq i \leq N_{WS_k}\right\} \equiv A \tag{5.10}$$

Kim and Park provide a simple algorithm for carrying out the re-ordering, although any method that allows the one-to-one mapping between indexes will suffice.

The *ith* element of $P_k(t)$ therefore represents the probability that the system is in state $\delta_k(i)$, at time *t* from the start of phase *k*. The state probability vector for the end of a phase *k*, $P_k(t_k - t_{k-1})$, is found from the phase's initial state probability vector, $P_k(0)$ through Equation 5.4. The reduced state probability vector for the start of the following phase is then formed by copying across the value from the end of phase reduced state vector to the element corresponding to the same system state. Elements in the next phase's probability vector corresponding to a system failed system state in the previous phase, and therefore not present in the previous phase's end of phase probability vector, are initialised to a value of 0. Another possibility is that an element in an end of phase probability vector does not represent a system failed state in that phase but does in the next phase, and is therefore not represented in its initial probability vector. Summing these elements gives the probability of transitional failure between the phases. The mission unreliability is then determined from the state probability vector for the end of the mission, $P_N(t_N - t_{N-1})$, and is calculated as 1 minus the sum of the elements of this vector, as shown in Equation 5.11, since these elements represent the success states for the final phase.

$$Q_{Miss} = 1 - \sum_{i=1}^{N_{WS_k}} P_{N,i}(t_N - t_{N-1}) \qquad \textbf{5.11}$$

5.1.2 Single Transition Matrix

An alternative to separate state transition matrices for each phase is to use a single state transition matrix for the complete mission. The main challenge with this approach is to avoid a huge state space that is computationally very expensive to solve.

Dugan [46] presents an alternative method for Markov analysis of phased missions which avoids the process of mapping end of phase state probability vectors to the start of phase state probability vectors for the following phase. A single state transition-rate matrix is formed with the full set of states for the components in the model but with transition rates that vary according to the current phase of the mission. The state transition matrix is therefore the same size as a state transition rate matrix resulting from the method by Alam and Al-Saggaf.

The variation in transition rates between phases is achieved by multiplying a transition rate for phase *j* by σ_j, where σ_j equals 1 during phase *j* and 0 at all other times. As in the other methods, the failure states specific to each phase are made absorbing, preventing the system from returning to a working state since the mission would have ended.

At the end of each phase, the probability of phase failure and failure on transition to the next phase are calculated from Equations 5.5 and 5.6. The probabilities of all failure states for that and the next phase are then set to 0, the transitions rates are switched to those of the next phase and the analysis for the next phase begins. Once the end of the mission is reached, the mission unreliability is then found through Equation 5.8.

The method therefore results in a single state transitional matrix of the same size as the phase state transitional matrix from Alam and Al-Saggaf's, whilst having only this matrix to solve compared to one for each phase. The

disadvantages are that the state space cannot be minimised for each phase as it can with Kim and Park's method and that a numerical solution scheme must be used leading to greater computational expense.

Smotherman and Zemoudeh [47] investigated a Markov modelling approach suitable for analysing repairable phased mission systems with time dependant failure and repair rates, random and state dependant phase transition times and multiple mission objectives. It is therefore able to model aspects of phased missions that are beyond the scope of the other Markov based methods found in the literature.

Their model includes a different Markov state for each system component state in each phase and for each mission objective. For example, with a two component system, there could be a state for both components working in phase one, a state for both components working in phase two with objective (or task) 1 and a state for both components working in phase two with objective (or task) 2, amongst others. The transitions between the different states are then determined and include, as well as the component failure and repair transitions, transitions for phase transitions (i.e. rates for the transfer between phases). These phase transition rates can vary between different Markov states; indeed the phase transition rate may be 0 from a certain state, if the system cannot progress to another phase from it. The system is assumed to be in the all component working state at the mission start time and the resultant differential equations in the model are then solved through a Runge-Kutta numerical solution scheme. To model the phase transition times, uniform probability density functions are used that are zero outside the range of possible transition times. This results in the rate of phase transition before the earliest transition time being zero, increasing and approaching infinity as the time reaches the latest transition time and being set to zero after the end of the interval. The phase transition time distribution intervals must be non-overlapping for multiple phase transitions from the state, for example when different phase transitions represent transitions to different objectives in the phase entered. The mission status probabilities, such as the probability a certain objective was reached or whether failure occurred, are found from the state probabilities at the mission end time, through Equation 5.8.

Although this method is able to analyse complex phased missions with state dependant transition rates, it can result in very large state spaces. The state space increases rapidly with the number of phases, unlike with the other Markov methods that either have a separate state transition rate matrix for each phase or a single state transition rate matrix with non phase specific system states. For this reason, the analysis of large systems with many phases using this method may be difficult due to limitations in available computational resources.

5.2 BDD based method

Recently, the BDD technique has become the preferred method for the analysis of non-repairable phased missions due to its ability to derive exact unreliability at relatively low computational expense [33]. Although the method has limitations when applied to the analysis of repairable systems, a method developed by Wang and Trivedi [48] that is discussed in this section demonstrates its potential for the efficient analysis of certain types of repairable phased mission.

Even with the recent developments in the Markov methods that result in smaller state spaces, the models can still present computational difficulties when large systems are analysed. Wang and Trivedi [48] presented a

method that uses the BDD technique to represent the system structure and continuous time Markov chains to model the state of individual components. The use of small Markov models for each individual component instead of modelling the complete system means that the maximum state space size is kept very small. The major advantage is that, since large state spaces are avoided, larger systems can be modelled efficiently. The main difference from the Markov modelling based methods is that the method assumes repaired components are only integrated back into the system at the end of a phase instead of immediately upon repair. This limits the types of phased mission system that can be modelled accurately. Unlike the phase dependencies for non-repairable systems that can be incorporated into the BDD construction procedure to minimise BDD size, such as by the algorithm presented by Zang, Sun and Trivedi [31], when repair is included the situation is more complex and the BDD cannot be minimised during construction. The method therefore assumes all events are independent for the purposes of BDD construction and the original fault tree to BDD computation operations from Rauzy [14], Equations 2.14 to 2.17, are therefore used. The dependencies between events on the paths through the BDD are then resolved during the probability evaluation. The authors present two node variable ordering schemes along with their corresponding evaluation procedures, each having its own advantages. The first optimises for memory requirements as it results in a smaller BDD size but an evaluation speed that is linear with the number of paths (which is normally many orders of magnitude greater than the number of nodes), whilst the second results in a larger BDD size but an evaluation speed that is linear with the number of nodes. The first scheme groups variables from the same phase together, with arbitrary ordering within a group, and variables of a later phase behind those from earlier phases across groups. The second scheme groups variables of the same component together, with variables of an earlier phase behind those from later phases within each group, and arbitrary ordering across groups.

Each component is modelled as an arbitrary, finite state, homogeneous continuous time Markov chain with a set of failed states and a set of working states, thus allowing components with multiple failure modes to be modelled at the component level.

The n states for component c can be split into two sets, where states $1,2\ldots,m$ are the operational states and states $m+1,m+2,\ldots,n$ are the failure mode states. An n by n matrix, $Q^{(C)}$, can then be formed, where element (i, j) is the transition rate from state i to state j of component c. The method therefore models components with multiple operational as well as multiple failure states. The matrix can be partitioned into four matrices as shown in Equation 5.12, where Q_{11} is an m by m matrix containing transitions from operational to operations states, and Q_{21}, Q_{12} and Q_{22} contains transitions from failed to operational states, operational states to failed states and failed to failed states, respectively.

$$Q^{(C)} = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \qquad \textbf{5.12}$$

Transition matrix $P^{(C)}$, shown in Equation 5.13, can then be formed by replacing Q_{21} and Q_{22} in $Q^{(C)}$ by zeros matrices of the same size, thus the difference between $P^{(C)}$ and $Q^{(C)}$ is that $P^{(C)}$ contains no transitions out of failed states (i.e. no repairs).

$$P^{(C)} = \begin{pmatrix} Q_{11} & Q_{12} \\ 0 & 0 \end{pmatrix} \quad \textbf{5.13}$$

The matrix $E_i^{(C)}$ where element (*j*, *k*) is the probability that component *C* ends phase *i* in state *k* given that it started the phase in state *j* can be found from the exponential of the matrix $Q^{(C)}$, as shown by Equation 5.14. Matrix $U_i^{(C)}$, where element (*j*, *k*) is the probability that component *C* ends phase *i* in state *k* given that it started the phase in state *j* and the component remained operational throughout the phase, is found from the exponential of matrix $P^{(C)}$ and is given by Equation 5.15, where $I_{m\times m}$ is the *m* by *m* identity matrix.

$$E_i^{(C)} = e^{Q^{(C)}(t_i - t_{i-1})} \quad \textbf{5.14}$$

$$U_i^{(C)} = e^{P^{(C)}(t_i - t_{i-1})} \cdot \begin{pmatrix} I_{m\times m} & 0 \\ 0 & 0 \end{pmatrix} \quad \textbf{5.15}$$

Finally, matrix $D_i^{(C)}$, where element (*j*, *k*) is the probability that component *C* ends phase *i* in state *k* given that it started the phase in state *j* and the component failed at some point during the phase, is found from matrices $E_i^{(C)}$ and $U_i^{(C)}$, as shown by Equation 5.16.

$$D_i^{(C)} = E_i^{(C)} - U_i^{(C)} \quad \textbf{5.16}$$

The state of component *C* in phase *i* is represented by a Boolean variable as shown by Equation 5.17, where a value of 1 for the variable indicates that component *C* fails at sometime during phase *i*, 0 indicates that component *C* is operational throughout phase *i* and *x* indicates that the status of component *C* during phase *i* doesn't matter.

$$b_i^{(C)} = \begin{cases} 1, \\ 0, \\ x, \end{cases} \quad \textbf{5.17}$$

The joint probability vector, where the k^{th} element is the probability that component *C* is in state *k* at the end of phase *p* given a set of mappings for the Boolean variables representing the state of *C* in phases 1 to *p*, is shown by Equation 5.18, where $v_0^{(C)}$ is the initial probability vector for component *C* and $X_i^{(C)}$ is defined by Equation 5.19.

$$P(b_1^{(C)} = s_1, b_2^{(C)} = s_2, ..., b_p^{(C)} = s_p) = v_0^{(C)} \cdot \prod_{i=1}^{p} X_i^{(C)} \quad \textbf{5.18}$$

$$X_i^{(C)} = \begin{cases} U_i^{(C)} & if\ s_i = 0, \\ D_i^{(C)} & if\ s_i = 1, \\ E_i^{(C)} & if\ s_i = x. \end{cases} \quad \textbf{5.19}$$

The joint probability can then be found from the joint probability vector, defined in Equation 5.18, through Equation 5.20, where 1^T is a unit column vector of size *m*.

$$P_r\{b_1^{(C)} = s_1, b_2^{(C)} = s_2, ..., b_p^{(C)} = s_p\} = P(b_1^{(C)} = s_1, b_2^{(C)} = s_2, ..., b_p^{(C)} = s_p) \cdot 1^T \quad \textbf{5.20}$$

Wang and Trivedi present efficient algorithms for evaluating the probability of a BDD node from a BDD constructed using each of the Boolean variable ordering schemes. As states earlier, although the BDD resulting from the second ordering scheme is larger, it can be evaluated far more efficiently. The evaluation procedure for this ordering scheme will now be described.

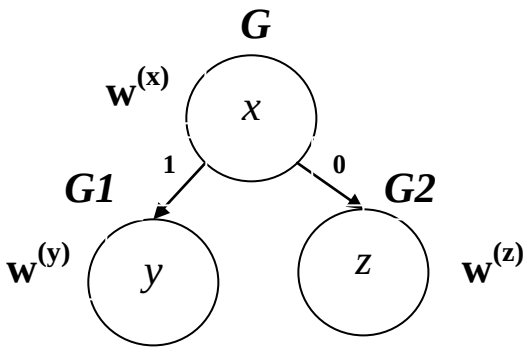


Figure 5.1 – BDD node *G* and its children, G1 and G0, labelled with their state probability vectors.

Figure 5.1 shows node *G* from a BDD with its failure and success child, *G1* and *G0* respectively, labelled with their respective node variables *x*, *y* and *z*. Here *x* represents a variable from component *c* in phase *i* and therefore, due to the ordering scheme, variables *y* and *z* must either represent variables from a different component or from the same component and a lower phase. The state probability vector $w^{(x)}$, where the *kth* element is the probability that the Boolean function *G=1* and component *c* is in state *k* at the end of phase *i*, is cached in node *G* and calculated through Equation 5.21.

$$w^{(x)} = p(G=1) = p(x=1, G_{x=1}=1) + p(x=0, G_{x=0}=1) \quad \textbf{5.21}$$

The state probability vector $p(x=1, G_{x=1}=1)$ in Equation 5.21 is calculated through Equation 5.22 if *x* and *y* both represent variables from component *c* in phases *i* and *j* respectively, where *j*<*i* due to the ordering, and through Equation 5.23 otherwise.

$$p(x=1, G_{x=1}=1) = w^{(y)} \cdot \left(\prod_{k=j+1}^{i-1} E_k^{(c)} \right) \cdot D_i^{(c)} \text{ if } cp(x) = cp(y). \quad \textbf{5.22}$$

$$p(x=1, G_{x=1}=1) = v_0^{(C)} \cdot \left(\prod_{k=1}^{i-1} E_k^{(c)} \right) \cdot D_i^{(c)} \cdot \left(w^{(y)} \cdot 1^T \right) \text{ if } cp(x) \neq cp(y). \quad \textbf{5.23}$$

The state probability vector $p(x=0, G_{x=0}=1)$ in Equation 5.21 is calculated through Equation 5.22 if *x* and *z* both represent variables from component *c* in phases *i* and *j* respectively, where *j*<*i* due to the ordering, and through Equation 5.25 otherwise.

$$p(x=0, G_{x=0}=1) = w^{(z)} \cdot \left(\prod_{k=j+1}^{i-1} E_k^{(c)} \right) \cdot U_i^{(c)} \text{ if } cp(x) = cp(z). \quad \textbf{5.24}$$

$$p(x=0, G_{x=0}=1) = v_0^{(C)} \cdot \left(\prod_{k=1}^{i-1} E_k^{(c)} \right) \cdot U_i^{(c)} \cdot \left(w^{(z)} \cdot 1^T \right) \text{ if } cp(x) \neq cp(z). \quad \textbf{5.25}$$

The state probability vector for each node can be calculated through Equations 5.21 to 5.25, starting from the bottom of the BDD with the terminal one and terminal zero nodes, whose state probability vectors are (1,0,...,0) and (0,0,...,0) respectively. The probability of the top event from the fault tree is then given by the sum of the elements in the state probability vector from the root BDD node. For full descriptions of each of the algorithms, including pseudo code implementations, see Wang and Trivedi [48].

Although modelled at the component level, at the structural level, multiple failure modes are not modelled, thus the occurrence of all failure modes are treated the same in the reliability structure. Due to this no advantage is gained from the detailed component failure modelling with the current structural model, although the authors intend to extend it to model multiple failure modes in future work.

Analysing the reliability of a phased mission through this method results in much lower computational expense than through the Markov methods discussed earlier. However, due to the assumption that repaired components are only integrated at the end of a phase, it is only suitable for the analysis of phased missions where this is the case.

5.3 Markov Computation

All techniques discussed in this literature review so far involve the use of Markov techniques. All the methods use the Markov technique to model the system except the method described in section 5.2 where it is used to model individual components. In this section the term *states* will be used to refer to the system or component states modelled by the Markov methods. The Markov techniques involve the solution of Equation 5.4 which describes the relationship between the probabilities that the system or component is in each state at an initial and some future time. The computational requirements for the solution of this equation in terms of both memory and processing power are discussed in this section.

The memory requirements are largely governed by two factors:

1. The order of the transition rate matrix - itself governed by the number of states represented.
2. The proportion of non-zero elements in the matrix.

The number of elements in the transition rate matrix, and therefore potentially the number of transition rates that must be stored in memory, increases with the square of the number of states represented. Fortunately, this matrix is usually sparse as direct transitions cannot occur between many pairs of states and this allows encodings that achieve greater storage efficiency to be used. A number of methods exist for computing the matrix exponential in Equation 5.4, whose exact value is given by a converging infinite power series, with varying degrees of complexity and accuracy, for more detail see Moler and Loan [49]. Iterative schemes for solving Equation 5.4 such as Runge-Kutta methods also exist, which compute the results by successive approximations using small time steps. These schemes rely on the choice of a time step that is small enough to give accurate results without being so small as to add unnecessary computational expense. Rauzy [50] discusses and compares several of these iterative schemes including advice on choosing a good time step value. The processing power required to compute a solution to Equation 5.4, i.e. the future state probability vector from the initial state probability vector and transition rate matrix, increases rapidly with the size of the transition rate matrix. For example, the computational resources required to calculate the exponential of an *n x n* matrix using the widely used Páde Approximation with scaling and squaring is *O(n^3)* [49]. Thus a 10 fold increase in the order of the transition rate matrix leads to the computational expense of its solution increasing by around 1000 times. The overall computation requirements for the use of the Markov methods, particularly in terms of processing power but also in terms of memory, therefore grow substantially as the size of the transition rate matrix increases.

5.4 Techniques for reducing computational expense

The Markov modelling methods can suffer from an explosion in the size of the state when used to analyse large systems, resulting in models that are difficult to solve with the available computational resources for the reasons discussed in the previous section.

Sharma and Bazovsky [51] give examples of techniques that can be used with the Markov techniques to reduce the size of system state spaces. Since these reduction methods only apply to system state Markov models, they are not applicable to the component state Markov models that appear in the method from Wang and Trivedi

discussed in section 5.2. The techniques involve forming truncated Markov models where the system states representing large number of component failures are omitted, thus reducing their size. The upper and lower bounds for mission unreliability can then be found from these reduced state space models. A special state is designated as the destination for failure transitions from states at the component failure count truncation threshold. Assuming that this state is a system working or failed state gives lower and upper bounds on the unreliability respectively. They also show that where possible, splitting the system fault tree into separate trees such that the individual trees are independent and components have the same inspection interval, allows the analysis to be carried out on a set of smaller Markov models. This is beneficial due to the reduction in the computational effort required to solve several smaller Markov models in comparison with a single but much larger model.

5.5 Monte Carlo Simulation based methods

Monte Carlo simulation has been used to analyse repairable phased missions and is particularly suited to modelling very large or complex systems with many dependencies, for which the Markov modelling methods are not suitable.

Altshcul and Hagel [52] discuss the application of simulation to analyse phased missions through the phase fault trees.

More recently, Chew, Dunnett and Andrews [53] developed a method to model repairable phased missions using a three layer stochastic Petri net and a Monte Carlo simulation solution scheme. The method can be used to model the reliability over maintenance free operating periods (MFOP). MFOP is a period or series of missions over which the system must be able to operate, free of restrictions or limitations, without maintenance. The three layers of the Petri Net consist of the Phase, Component and Master Petri nets, which interact with one another through arcs to form a single Petri net. The Phase Petri nets model the system reliability structure in terms of component failure combinations for each phase of the mission, the Component Petri nets model the failure and repair of each component including the modelling of dependencies between components, whilst the Master Petri net models phase sequence and progression. The inclusion of component dependencies and state dependant phase transitions allows complex phased mission scenarios to be modelled. Monte Carlo simulation is used as the solution technique to determine reliability metrics from the model.

5.6 Summary and Conclusions

A number of methods have been developed to analyse the reliability of repairable phased missions. It was found that the Markov modelling technique has been most widely used. Alam and Al-Saggaf developed a method using state transition rate matrices for each phase, which was further developed by Kim and Park to reduce the size of the state space, thus improving computational performance. Dugan developed a method using a single state transition rate matrix with phase varying transition rates, removing the need for state probability vectors to transfer between phases. The disadvantages with this approach are that it prevents phase by phase optimisations of state space size and requires the use of a numerical solution scheme. A further Markov approach using a single state transition rate matrix was introduced by Smotherman and Zemoudeh that is able to model additional

aspects of phased missions such as state specific phase transition rates. However, since the same component states in different phases are treated as different states in the Markov model, the resultant state space can be huge when large systems with many phases are modelled. Wang and Trivedi used the BDD technique in a method that can analyse certain repairable phased missions with very low computational expense. For missions where repaired components are integrated back into the system at the phase end times then this method is much more efficient than the approaches based on Markov models. Chew, Dunnet and Andrews developed an approach that uses a multi layer Petri net and Monte Carlo simulation to analyse system reliability in maintenance free operating periods (MFOP).

6 New Methods for the analysis of repairable phased missions

6.1 Introduction

The literature review showed that a variety of methods have been developed for analysing the reliability of repairable phased mission systems. Amongst those based on Markov modelling, the method from Kim and Park [45] is the most efficient. Only the method by Wang and Trivedi [48], which uses both Markov and BDD techniques, has lower computational expense. However all the methods have far higher computational expense than the methods developed for non-repairable phased missions, especially the Markov only methods as they suffer from huge state spaces when large systems are analysed. All the reviewed methods have restrictions on the type of system that can be analysed, for example the method by Kim and Park is limited to those containing single failure mode components whilst the method by Wang and Trivedi assumes repairs are integrated at phase end times.

Due to the disadvantages of existing methods discussed above the research focus was on their modification and extension as well as using ideas developed in the field of non-repairable systems to create methods capable of analysing systems of the high complexity and large size often found in the real world.

The research led to the development of the following five methods:

1. An extension of the method from Kim and Park to the analysis of systems with multiple failure mode components.
2. A BDD based method for identifying which system states are system failure states.
3. An improved efficiency evaluation scheme for the BDD based method from Wang and Trivedi for repairable systems.
4. A method for the efficient analysis of systems containing both repairable and non-repairable components and where repaired components are integrated back into the system at the end of a phase.
5. A method for the efficient analysis of systems containing both repairable and non-repairable components where repaired components are reintegrated into the system immediately.

6.2 Method for analysing missions with multiple failure mode components

The Markov based repairable phased mission analysis method developed by Kim and Park [45] is the most efficient method for the analysis of repairable systems with immediate repair integration due to the phase by phase minimisation of the state space. However, that method is only described for systems containing single failure mode components and not those containing multiple failure mode components. An extension to the method that allows it to be used to analyse systems containing mutually exclusive failure mode components was therefore developed and is described in this section. Since many components have multiple failure modes, this method is applicable to a wider range of systems than the original from which it was extended.

Similar to the method from Kim and Park, the new method utilises the phase by phase state space optimisations to minimise computational expense. As shown in the literature review, this relies on the identification of the system states representing system failure. The analysis of multiple failure modes results in larger state spaces and the availability of an efficient method for this step is therefore particularly important for reducing computational expense. For this reason an efficient technique for performing this step has also been developed.

The extension of Kim and Park's Markov method to systems containing multiple failure mode components and the efficient Markov state evaluation method are presented in the following sections.

6.2.1 Extending the Markov method to systems with multiple failure mode components

The method from Kim and Park, reviewed in section 5.1.1, is extended for the analysis of systems with multiple failure mode components in this section. Only small changes to some of the definitions are necessary due to the change in the state space size caused by multiple failure mode components, whilst the overall methodology remains unchanged.

The first step is to determine the exhaustive set of Markov states that each represents a different system state, i.e. the set containing a state for every possible combination of states for the system's components. Since failure modes are mutually exclusive, no system state can have more than one failure mode from the same component present. The total number of Markov states, N_{ms}, is therefore given by Equation 6.1, where N_c is the number of components represented in the phase fault trees for the mission to be analysed and N_{fm_i} is the number of failure modes for component *i*. The equation shows that the size of the state space can increase very rapidly when the system contains components with many failure modes.

$$N_{ms} = \prod_{i}^{N_C} (N_{fm_i} + 1) \qquad \textbf{6.1}$$

A N_{ms} by N_{ms} transition-rate matrix, β, is then formed containing the transition rates between these system states. The element at row *i* and column *j* of β, $\beta(i, j)$, for $i = 1..N_{ms}$ and $j = 1..N_{ms}$ and $i \neq j$ is set to the rate of transition from state *j* to state *i*, $\lambda_{j,i}$, as it was in the method from Alam and Al-Saggaf [44]and as was shown in Equation 5.1.

Each element on the diagonal of β, $\beta(i,i)$, for $i = 1 \ldots N_{ms}$, is set to the negation of the sum of the other elements on column *i* and represents the transition rate out of state *i*, as shown in Equation 6.2. Again this is the same as in the method from Alam and Al-Saggaf but with a range of $i = 1 \ldots N_{ms}$ rather than $i = 1 \ldots 2^{N_C}$.

$$\beta(i,i) = -\sum_{j=1,\, j \neq i}^{N_{ms}} \beta(j,i) \qquad \textbf{6.2}$$

The remainder of the method remains unchanged from Kim and Park, as was described in section 5.1.1, except for the use of a state probability vector for time *t* in phase *k*, $P_k(t)$, of length N_{ms} instead of 2^{N_C}. The initial state probability vector, $P_1(0)$, used in the method will have a value of 1 for the element representing the state for the absence of any failure modes for all components and a value of 0 for all others.

6.2.2 Using BDDs to identify system failure states

In this section a method is presented that utilises the BDD technique to very efficiently determine whether a particular system state, i.e. set of component failure modes that are present, represents a failed or working system state in a particular phase. Such a method is required to form the correct reduced state transition rate matrix for a phase from the full state transition rate matrix but none is given by Kim and Park. Efficient state evaluation is critical to the methods overall efficiency since the number of system states that must be evaluated for each phase is 2^{N_C} in Kim and Park's method and N_{ms} in the method given in section 6.2.1. For example, a system consisting of 10 components with 2 failure modes each and operating in a mission of 5 phases would require the evaluation of 295,245 system states. This efficiency is of even greater importance in the method for systems containing both repairable and non-repairable components, described later in section 6.4.2, where multiple reduced state matrices may need to be formed for the same phase.

6.2.2.1 BDD Construction

In this method, a separate BDD is constructed from each phase fault tree in order to evaluate whether the system is failed for each system state in that phase. These BDDs are constructed using the rules from Rauzy that assume node variable independence, given in section 2.5.4.3, except when computing the result of a logical operation between nodes with Boolean variables from different failure modes belonging to the same component. In that case rule 2 from Table 4.1 is used. An example fault tree is shown in Figure 6.1a and its BDD equivalent, constructed according to these rules, is shown in Figure 6.1b.

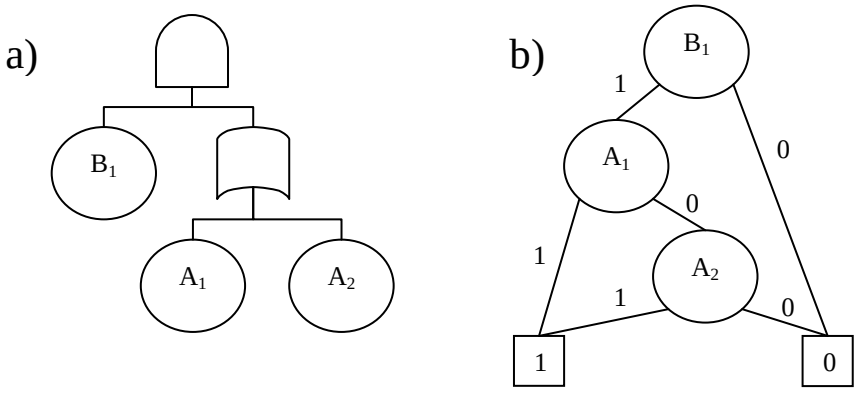


Figure 6.1 – An example fault tree and its BDD.

6.2.2.2 Markov State Evaluation

To evaluate the system status (working or failed) for a certain system state and phase, e.g. to evaluate if a particular Markov state in the method from Kim and Park can be removed from the state transition rate matrix, the BDD for that phase (constructed according to the rules given in the previous section) is used. The algorithm for checking whether the system is failed for a particular system state is very simple. The top node of the BDD is entered and the failure mode of its Boolean variable is checked against the set of component failure modes present in the system state. If the failure mode exists in the system state then the BDD node's failure child is evaluated against the system state next, otherwise it is the success child. This continues down the BDD until a terminal node is reached, the value of which indicates whether the system state being considered is a failed or working state in that phase. Specifically, reaching the BDDs terminal 1 node indicates that the system state represents system failure whilst reaching the terminal 0 node indicates that it represents system success. The key factor in the efficiency of this method is the systematic elimination of comparisons for failure modes that are non-critical to the system outcome due to the removal of nodes with isomorphic children during BDD construction. At each step in the process only a failure mode whose presence can influence the terminal node reached (i.e. whether or not the system state represents system failure) is checked, based on the outcome of the failure mode comparisons previously made in the higher nodes. The maximum number of evaluations required to determine the system status for any system state is therefore equal to the maximum depth of the BDD (i.e. the maximum number of nodes on any route through the BDD from the root to a terminal node). A pseudo code implementation of the evaluation procedure is shown below:

```
evaluate_markov_state(ss, t_node):
# ss is the system state to be evaluated.
# t_node is the top node of the BDD used to evaluate the system state.
# evaluate_markov_state(ss, t_node) returns true if the system state represents system failure based on the BDD,
# and false otherwise.
if t_node == terminal_one_node:
        return true
else if t_node == terminal_zero_node:
        return false
else if failure_mode(t_node) in ss:    # evaluates to true if the variable of t_node belongs to a failure mode in ss
        t_node = failure_child(t_node)        # sets t_node to its failure child.
        return evaluate_markov_state(ss, t_node)
else:
        t_node = success_child(t_node)      # sets t_node to its success child.
        return evaluate_markov_state(ss, t_node)
```

For example, the system state representing the occurrence of failure mode 1 of component *A* would enter the top node of the BDD shown in Figure 6.1b where the presence of failure mode 1 of component *B* would be checked, since this is the variable for the top node. Since it is not present the traversal would continue to the success son, which in this case is the terminal 0 node, thus indicating that the system state represents system success.

6.2.3 Summary

Kim and Park's Markov based method for analysing repairable systems is limited to the analysis of single failure mode component systems and requires the classification of component state combinations into the system state they represent. No method for this had been previously described. The methods described in this section:

- Extend that method to the analysis of systems with multiples failure mode components.
- Efficiently determine which component state combinations represent system failure states.

They have therefore extended the range of systems that can be analysed using these Markov based methods and provide an efficient algorithm to be used as part of their implementation.

6.3 Improved Evaluation Efficiency in Wang and Trivedi BDD method

The method presented in this section is an alternative to the BDD probability evaluation algorithm given by Wang and Trivedi [48] in their method discussed in section 5.2. This new evaluation procedure results in improved computational efficiency and therefore faster evaluations. The feasibility of applying a phased mission reliability evaluation technique to certain types of analysis, such as real time or importance measure, is dependent on its computational expense. If the technique has high computational expense, then the time taken to compute results from the analysis will be too great, particularly when large or complicated phased missions are analysed. Improvements in efficiency to existing techniques, such as those introduced in this section, are therefore useful in those contexts.

Two sources of inefficiency were identified in the algorithm given by Wang and Trivedi and these will now be explained. A BDD node can, and often does, have multiple parents in a BDD (due to the BDD's sharing of isomorphic sub graphs property) and more than one of those parent nodes may have a variable belonging to a different component than its variable. Since, in the probability calculation method given by Wang and Trivedi, the probability vector from a child node, $w^{(y)}$ or $w^{(z)}$, is summed in the parent node, if the nodes have variables belonging to different components then there is an inefficiency as the same vector may be summed multiple times. This inefficiency occurs each time the probability of the BDD is evaluated. The second inefficiency, which occurs only on repeat probability evaluation of the same BDD, is the repetition of the comparisons between the component of the node's variable and that of its failure and success children. The two inefficiencies identified represent only a small proportion of the overall computation in the method since the computation of the matrix exponentials has higher relative expense. However, even small decreases in the probability calculation time due to increased efficiency are valuable in applications such as real time analysis.

Research into the evaluation procedure led to the development of a new method that removes the identified inefficiencies. The changes involve the processing and conversion of the BDD into a different data structure and an evaluation procedure that operates on that new data structure instead of the BDD. The source of the improved efficiency is the avoidance of the repetition of processing that is performed during the BDD evaluation in the

original method. Instead that processing is performed just a single time during the conversion from BDD to the new data structure. The new data structure created is similar to the Implicant Tree data structure that was discussed in section 4.2.2.2 and is named the Repairable Implicant Tree. The maximum benefits in terms of increased efficiency are seen when the Repairable Implicant Tree is used to re-evaluate the probability of the same reliability structure, e.g. during real time or importance measure analysis.

Other than the BDD probability evaluation, the rest of the method (such as BDD construction) remains the same as given by Wang and Trivedi.

6.3.1 BDD Evaluation through Repairable Implicant Tree

In this section the details of the Repairable Implicant Tree data structure, its construction from the BDD and its probability evaluation are explained. As with the non-repairable Implicant Tree, a node in the structure represents a set of mutually exclusive implicants whose probability can be calculated from the probability of the node's event and the probabilities of its immediate child nodes.

6.3.1.1 Structure

The Repairable Implicant Tree consists of two types of node, named Event nodes and Terminal nodes, which are described in this section.

Event Node

An Event node represents an *ite* node from the BDD. Unlike an *ite* node, the edges to its child nodes can be of one of two types, named a Vector edge and an Absolute Value edge, the significance of which is explained in section 6.3.1.2.

Terminal Node

A Terminal node represents a terminal one or terminal zero node from the BDD.

6.3.1.2 Construction

This section explains the construction of the Repairable Implicant Tree from a BDD that was constructed assuming node variable independence and using the second ordering scheme given by Wang and Trivedi. The construction process eliminates the need to compare node variables on repeat evaluations and limits the summing of a probability vector to a single time on each evaluation. By using the two types of edge there is no need to determine whether the child node belongs to the same component or not during repeat evaluations, that knowledge is built directly into the data structure during its construction.

The Repairable Implicant Tree for a particular BDD node, defined as Repairable Implicant Tree *R*, is formed through the procedure outlined below:

If the BDD node is a terminal one or terminal zero node, then *R* is the corresponding terminal node since they represent the certain and impossible events respectively. Otherwise the BDD node must be an *ite* node.

R is formed from steps *a*, *b*, *c* and *d* described below:

a. The set of implicants represented by an *ite* node is a combination of the set of implicants represented by its child node combined (through a logical AND) with the failure and success events for the *ite* node's variable. The first step is therefore to form the Implicant Tree nodes representing the child BDD nodes, which are defined as Repairable Implicant Trees *U* and *V* respectively. *U* and *V* are also formed from the procedure outlined here and this is therefore a recursive process that ends at the terminal nodes of the BDD.

b. Create an Event Node to represent the *ite* BDD node. This has an edge to each of the child Implicant Tree nodes, *U* and *V*, which were created in step *a).* Unlike the BDD *ite* nodes which have only one type of edge, here the type of each edge varies depending on the component of the child node's variable to which it links. The node therefore holds additional information on its relationship with its child node, allowing higher evaluation efficiency. For each of the child nodes:
    - If its variable belongs to the same component as the *ite* node, component *C*, then it is connected via a Vector edge since this node's probability calculation will then require the probability vector from the child node.
    - Otherwise it is connected by an Absolute Value edge, including when it is a terminal node, since this node's probability calculation will only require the sum of the probability vector from the child node.

In steps *c* and *d* the failure and success event probability matrices are calculated and stored in the node. These are later used in the evaluation procedure, together with the child node probabilities, to determine the probability for this node.

c. Calculate the failure event probability matrix, $X_{m,i}^{(c)}$, given by Equation 6.3, where *i* is the phase of the BDD node's variable, *m* is the phase of the BDD node's failure son's variable if it also belongs to component *C* or 0 otherwise, and $E_k^{(c)}$ and $D_i^{(c)}$ are as defined in Equation 5.19. Element *(p,q)* of $X_{m,i}^{(c)}$ is the probability that component *C* ends phase *i* in state *q* given it ended phase *m* in state *p* and the component fails in phase *i*. Thus, the probability of component *C* ending phase *i* in each of its states given it fails in phase *i* can be calculated from this matrix together with the probability vector or value obtained from Implicant Tree node *U* during evaluation.

$$X_{m,i}^{(c)} = \left(\prod_{k=m+1}^{i-1} E_k^{(c)} \right) \cdot D_i^{(c)} \qquad \textbf{6.3}$$

d. Calculate the success event probability matrix, $Y_{n,i}^{(c)}$, given by Equation 6.4, where *i* is the phase of the BDD node's variable, *n* is the phase of the BDD node's success son's variable if it also belongs to component *C* or 0 otherwise, and $E_k^{(c)}$ and $U_i^{(c)}$ are as defined in Equation 5.19. Element *(p,q)* of $Y_{n,i}^{(c)}$ is the probability that component *C* ends phase *i* in state *q* given it ended phase *n* in state *p* and

the component does not fail in phase *i*. Thus, the probability of component *C* ending phase *i* in each of it states given that it works throughout phase *i* can be calculated from this matrix together with the probability vector or value obtained from Implicant Tree node *V* during evaluation.

$$Y_{n,i}^{(c)} = \left(\prod_{k=n+1}^{i-1} E_k^{(c)} \right) \cdot U_i^{(c)} \qquad \textbf{6.4}$$

An *ite* BDD node whose variable belongs to component *C* in phase *i* is therefore represented by the Repairable Event Node shown in Figure 6.2.

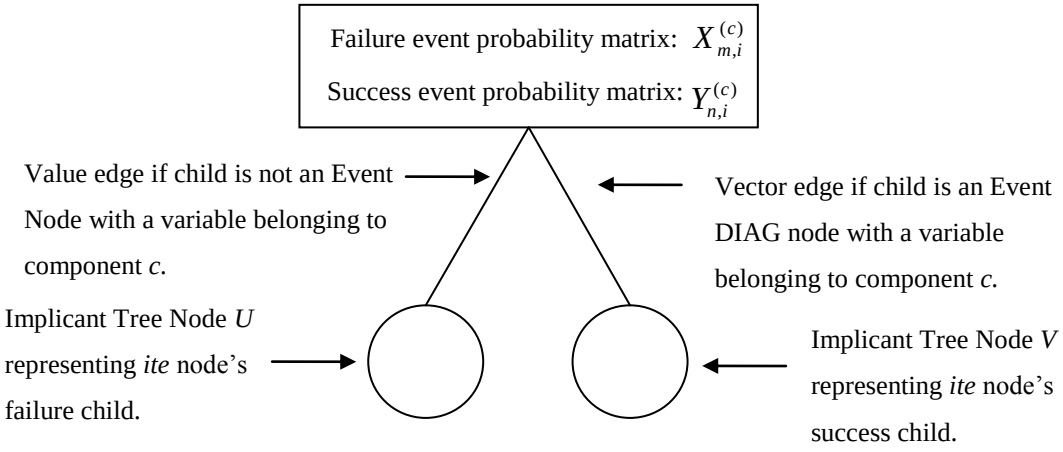


Figure 6.2 – Diagram showing the Event Node representation of a BDD *ite* node whose variable belongs to component *C* in phase *i*.

Since the failure and success event matrices calculated in steps *c* and *d* depend on the failure and repair rates for component *C* between phase *i* and phase *k* and the durations of those phases, the matrices must be updated if any of those variables change between evaluations. Note that only the affected matrices would need to be updated, whilst the others would remain constant; it is for this reason that the matrices are pre-computed during construction rather than during the execution of the evaluation procedure.

6.3.1.3 Evaluation

With the Implicant Tree data structure constructed it remains to specify the evaluation procedure used to quantify the probability. Due to the properties of that data structure many of the tasks that are normally performed when evaluating directly from the BDD, such as comparing node variables, have been eliminated. The evaluation is a recursive process that requires the calculation of a state probability vector and absolute probability value for each node in the Implicant Tree. It is initiated at the root node and commences in a bottom up manner with the probability of a terminal node evaluated first. The overall probability for the Implicant Tree

data structure, and hence the BDD from which it was constructed, is then given by the absolute probability value of its root node.

A description of the evaluation procedure will now be given:

The process for calculating for an Event Node is given by steps *a* and *b* below, whilst for a Terminal Node it is given by step *c*:

a. Obtain state probability vectors from node *U*, the failure child, and node *V*, the success child. If the child is connected with an Absolute Value edge then its absolute probability value is obtained and converted into a state probability vector, w_{child}, through Equation 6.5, where $v_0^{(C)}$ is the initial state probability vector for component *C*, the component to which the Event Node belongs, and P_{child} is the probability of the child node. If it is instead connected with a Vector edge then its state probability vector, w_{child}, is obtained directly. The state probability vectors for child nodes *U* and *V* will be referred to as $w_{failure}$ and $w_{success}$ respectively. Note that no node variable comparisons take place in this step, unlike Wang and Trivedi's procedure where the node's variable is compared to each child node's during each evaluation.

$$w_{child} = v_0^{(C)} \cdot P_{child} \qquad \textbf{6.5}$$

b. The state probability vector for the Event Node, w, is then calculated through Equation 6.6 using the state probability vectors obtained in step *a* and event probability matrices $X_{m,i}^{(c)}$ and $Y_{n,i}^{(c)}$ that were computed during the construction process. The absolute probability value is then calculated as the sum of the elements of vector *w*. Both the state probability vector and absolute values are cached within the node after calculation, ensuring that multiple parent nodes can use them without recalculation. This avoids the repeated summing of the state probability vector in the parent nodes where a child node is shared due to merging of isomorphic sub graphs during BDD construction.

$$w = X_{m,i}^{(c)} \cdot w_{failure} + Y_{n,i}^{(c)} \cdot w_{success} \qquad \textbf{6.6}$$

c. The probability evaluation procedure for a terminal node simply returns the absolute value of 1 if it is a terminal one node or 0 if it is a terminal zero node, since they represent the certain and impossible events respectively.

When the Implicant Tree is re-evaluated then the cached values that were calculated in *b* above must be cleared and the evaluation procedure repeated.

6.3.2 Summary

Research discovered certain repeated calculations and node variable comparisons in Wang and Trivedi's phased mission repairable system analysis method that could be removed in order to boost efficiency. This involved the use of a new data structure and evaluation procedure that has efficiency advantages over the original method. It is particularly optimised for the case of repeat probability evaluations of the same reliability structure, such as those required in the case of real time analysis.

6.4 A method for analysing repairable phased mission systems that contain some non-repairable components

No research into methods for the analysis of phased mission systems containing both repairable and non-repairable components was found in the literature. Only methods where all components are considered to be either non-repairable or repairable have been developed. This meant that mixed systems had to be analysed through methods developed for repairable phased missions with the repair rate set to zero for the non-repairable components. However this approach is not ideal since the repairable methods have far higher computational expense than the non-repairable methods due to the additional complexities that repair modelling introduces.

Due to this, research was carried out into the development of methods that could analyse systems containing both types of component more effectively. These methods combine the strengths and developments from both non-repairable and repairable system reliability analysis and are presented in this section.

6.4.1 End of phase repaired component integration method

The method developed in this section can be used to analyse the probability of any phased mission fault tree for a system with both repairable and non-repairable components. It assumes that repaired components are integrated back into the system at the end of the phase in which their repair is completed (for example if a repair commences in phase 1 and is completed at some point during phase 3, then it will be reintegrated into the system at the end of phase 3 and ready for the start of phase 4). It also assumes that only the working and failed states of the repairable components (and not individual failure modes) are represented in the system's reliability structure. For the non-repairable components, mutually exclusive failure modes are fully modelled, at both the component and system reliability levels. The method is based upon the methods developed for non-repairable phased missions in chapter 4 and for repairable phased missions by Wang and Trivedi in [48], covered in section 5.2 of the literature review, both of which use the BDD technique. It results in a single BDD that represents the Boolean function for the system's reliability structure from which mission unreliability and phase failure probabilities can be calculated at low computational expense.

6.4.1.1 Construction of the BDD

The phased mission fault tree to be analysed is first transformed into a BDD. The BDD formed in this method isolates the non-repairable and repairable parts of the systems such that the children of nodes in the BDD with variables belonging to repairable components have variables belonging only to repairable components. To ensure this is the case, the BDD variables are ordered according to the following hierarchy: non-repairable

components < repairable components, ordered by component, ordered by failure mode (only represented in node variables for non-repairable components) and backward phase ordering (higher phase < lower phase), where $x<y$ signifies that x appears higher than y in the BDD. For example, a system with repairable components A and B, and non-repairable two failure mode components C and D, when operating in a two phased mission could have its Boolean variables ordered as follows: $C_1(0,2) < C_1(0,1) < D_2(0,2) < D_2(0,1) < A(0,2) < A(0,1) < B(0,2) < B(0,1)$.

The BDD is then built using the non-repairable method developed in Chapter 4. For the Boolean operations between nodes, the rules from Table 4.1 are used except when both nodes have variables belonging to repairable components. In that case, the Boolean operations that assume variable independence are instead used, as described in section 2.5.4.3.

6.4.1.2 BDD Evaluation through Implicant Tree

The probability evaluation process that gives the probability of the BDD is described in this section. The BDD is first converted into an Implicant Tree data structure and the probability evaluation is then performed on this.

As with the Implicant Tree data structures for a non-repairable BDD described in section 4.2.2.2 and a repairable BDD described in section 6.3.1, the Implicant Tree consists of a single root node, a single terminal node and a set of intermediate nodes. In this case the intermediate nodes consist of both the types described in section 4.2.2.2 and those described in section 6.3.1.1. Nodes in the BDD belonging to non-repairable and repairable components are converted into their Implicant Tree equivalents using the construction methods given in sections 4.2.2.2. and 6.3.1.2 respectively and evaluated through the rules given in sections 4.2.2.2 and 6.3.1.3 respectively.

6.4.1.3 Computational Advantage

Compared to using the BDD method from Wang and Trivedi to analyse the mission reliability of a system containing both repairable and non-repairable components, the method described above has much lower computational expense. The two sources of the computational advantage are described in this section.

6.4.1.4 Smaller and optimised BDD

The upper parts of the BDD, consisting of nodes with variables belonging to non-repairable components, are minimised using the BDD construction algorithm developed for non-repairable systems that was described in Chapter 4. In contrast, the lower parts of the BDD, consisting of nodes with variables belonging to repairable components, are constructed assuming variable independence since the dependencies from repair prevent the minimisations from being applied. If the repairable method from Wang and Trivedi were used, the complete BDD would be constructed assuming variable independence resulting in a far larger BDD when many of the components in the system modelled are non-repairable. To demonstrate this, the BDDs for the phased mission shown in Figure 6.3, which contains both a repairable and a non-repairable component, produced from each of the methods are shown in Figure 6.4. Figure 6.4a shows the BDD produced from the method for repairable systems by Wang and Trivedi, and contains 8 *ite* nodes. Figure 6.4b shows the BDD produced from the method

introduced in this section which contains just 3 *ite* nodes, a reduction of over 60%. These reductions in size contribute to the increased computational efficiency.

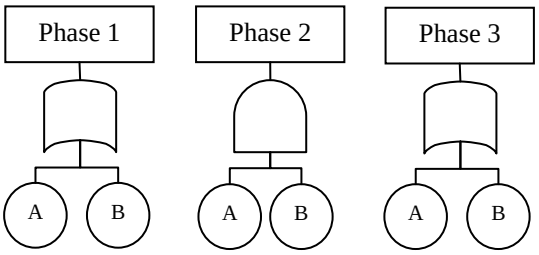


Figure 6.3 – Three phased mission for system with non-repairable component *A* and repairable component *B*.

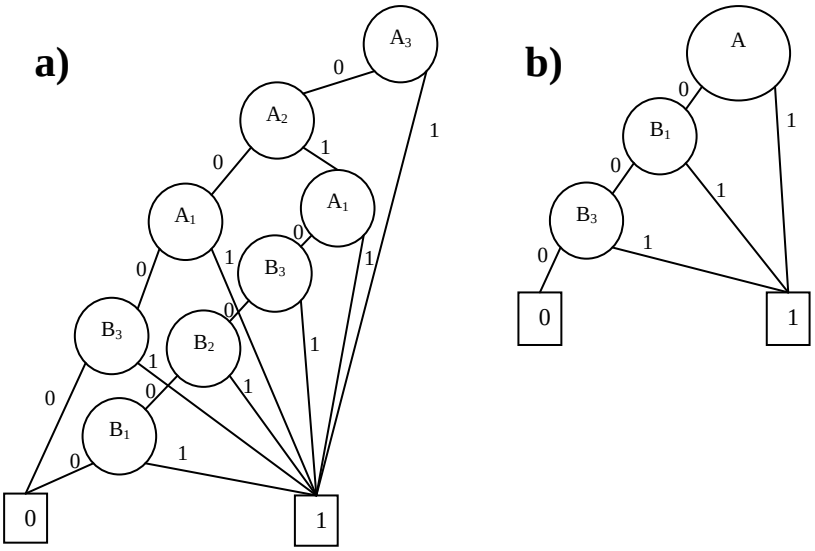


Figure 6.4 - a) BDD from Wang and Trivedi method. b) BDD from method optimised for systems containing both repairable and non-repairable components.

Faster evaluation

Unlike the Markov based methods that model system states, the Markov modelling of components in Wang and Trivedi's method does not suffer from the same huge computational expense caused by large transition rate matrices that was discussed in section 5.3, since the state spaces are generally far smaller. However, the use of the Markov technique to model any non-repairable components still adds unnecessary expense to the evaluation of the path probabilities compared to finding their failure probabilities directly from their failure distribution equations. In comparison, the new method, by modelling both repairable and non-repairable components in their optimal ways, considerably reduces the computational expense involved in evaluation when the system contains non-repairable components. For example, consider the evaluation of the parts of the paths containing non-repairable component *A* within the BDDs shown in Figure 6.4. The evaluation of the BDD shown in Figure 6.4a

with Wang and Trivedi's method involves the evaluation of 8 matrix exponentials and several matrix products, whereas the evaluation of the BDD in Figure 6.4b with the new method requires only the solution of a single simple equation for the exponential probability distribution.

6.4.1.5 Summary

A new method for the analysis of phased mission systems with both repairable and non-repairable components has been introduced. This new method has two advantages over the BDD method for repairable systems by Wang and Trivedi when systems containing both repairable and non-repairable components are analysed. The first advantage is that it has far lower computational expense, the magnitude of which becomes greater with higher proportions of non-repairable components. The second advantage is that it allows multiple failure mode components to be modelled at the system level for the non-repairable components, thus broadening the range of systems that can be analysed. Compared to the purely Markov modelling based methods, such as the method by Kim and Park [45], this BDD based method has far greater efficiency although it assumes that repaired components are integrated back into the system at the end of the phase rather than immediately.

6.4.2 Method for repairable components with multiple failure modes

The method introduced in this section is again suited to the analysis of the mission reliability of a phased mission system that has both non-repairable and repairable components. Unlike the method described in the previous section, it can model multiple component failure modes at the system level for both component types. It also differs in that it assumes that repaired components are integrated back into the system immediately rather than at the end of the phase in which their repair is completed. The method gives an upper bound for the mission unreliability and is therefore best suited to analysis where both:

a. The system is too large to model using one of the techniques that give exact results.
b. A conservative estimate of the mission reliability is all that is needed, e.g. to determine whether the mission is within certain unreliability constraints.

6.4.2.1 BDD Construction

The phased mission fault tree to be analysed, which for this method is limited to the mission fault tree describing failure at any point in the mission (i.e. an OR top gate with inputs of the phase fault trees), is first converted into a BDD like structure.

Node Types

A non-repairable *ite* BDD node has a failure child, a success child and a variable belonging to a component, failure mode and phase. A repairable *ite* BDD node has a failure child, a success child and a variable belonging to a component and failure mode. In addition to these standard non-repairable and repairable *ite* nodes, the structure also contains a special type of node named the Repairable Transition node. This is used to mark the transition in the structure between nodes with variables belonging to non-repairable and repairable components. The edges out of a Repairable Transition node, of which there can be 1 to *m*, where *m* is the number of phases in

the mission, are labelled with a phase number. The presence of this node type means that the structure is not strictly a BDD as some non-terminal nodes do not represent a binary decision.

Phase Fault Tree Basic Event to BDD node Conversion

A basic event for failure mode *j* of component *I* in the phase *k* fault tree is converted into the *ite* node shown in Figure 6.5a if component *I* is non-repairable or as shown in Figure 6.5b if it is repairable. As shown in Figure 6.5b, a basic event for a repairable component is represented by a Repairable Transition node with an *ite* node child, whose variable belongs to the failure mode and component of the basic event, linked by an edge labelled with the phase of the basic event.

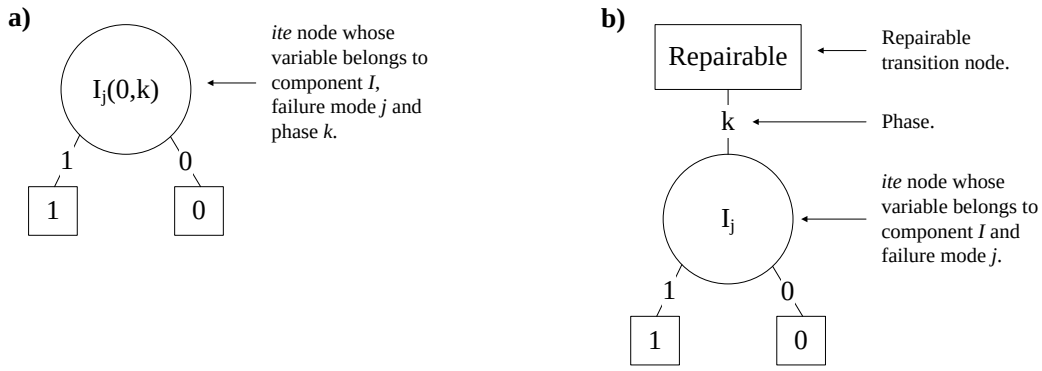


Figure 6.5 – BDD representations of basic events for non-repairable and repairable components.

Node Ordering Scheme

In this method the ordering scheme first orders nodes with variables belonging to non-repairable components < Repairable Transition nodes. There are then separate ordering schemes for ordering between node variables belonging to non-repairable components and between variables belonging to repairable components. Variables belonging to non-repairable components are ordered through the scheme given in section 4.2.1.1 whilst variables belonging to repairable components are ordered by component and then by failure mode (the difference is therefore that the phase of a variable belonging to a repairable component is not used).

Boolean computations between nodes

The computation of the result from a Boolean operation between two non-terminal nodes can be classified into one of the four cases listed below:

1. Between two non-repairable *ite* nodes.
2. Between non-repairable node and Repairable Transition node.
3. Between two Repairable Transition nodes.
4. Between two *ite* repairable nodes.

The rules given in section 4.2.1.2 are used to determine the result of the computation in cases 1 and 2.

Case 3, can be split into two sub cases:

a. Computation between Repairable Transition nodes that each has a single child node linked with an edge belonging to the same phase.
b. Computation between Repairable Transition nodes with child nodes linked with edges belonging to distinct phases. This is always a Boolean OR due to the restriction on only analysing phased mission fault trees representing mission failure.

For case 3*a*, the computation results in a new Repairable Transition node that has a child node, linked by an edge with the same phase as the edges from the input nodes, formed from the result of the Boolean operation between the child repairable *ite* nodes of the input nodes (see case 4 below). In case 3*b* the computation results in a new Repairable Transition node whose children are the union of the children from the two input nodes.

For case 4, the result is computed using the rules given in section 6.2.2.1.

For systems containing both repairable and non-repairable components this will result in a BDD like structure where the higher nodes are composed like a non-repairable BDD and then may make the transition, through a Repairable Transition node, into a set of phase repairable BDDs before the terminal nodes are reached.

BDD Minimisation

The set of mutually exclusive event combinations for non-repairable components represented by the paths from the top node to a Repairable Transition node, are a set of combinations for which the repairable component states that cause system failure in each phase are the same. These repairable component states are represented, for each phase, by the phase BDDs under the Repairable Transition node. Each of those phase BDDs contains only nodes with variables from repairable components whose states can affect the outcome of the mission, given that one of the mutually exclusive non-repairable event combinations occurs. This also means that a BDD for a certain phase will not be present under the repairable transition node if, based on the occurrence of one of the non-repairable component event combinations, failure cannot occur in that phase. The number of repairable transition nodes and the size of the phase by phase BDDs under each of them are therefore minimised by accounting for the relationship between the phase states of the non-repairable components, the phase states of the repairable components and the outcome of the mission.

Example BDD

The BDD for the example phased mission fault tree from Figure 4.10 is shown and annotated in Figure 6.6. In this mission components *D*, *E* and *F* are repairable. The BDD in Figure 6.6 was formed using an ordering scheme that ordered components as $B < A < C < D < E < F$ and ordered failure modes as $2 < 1$.

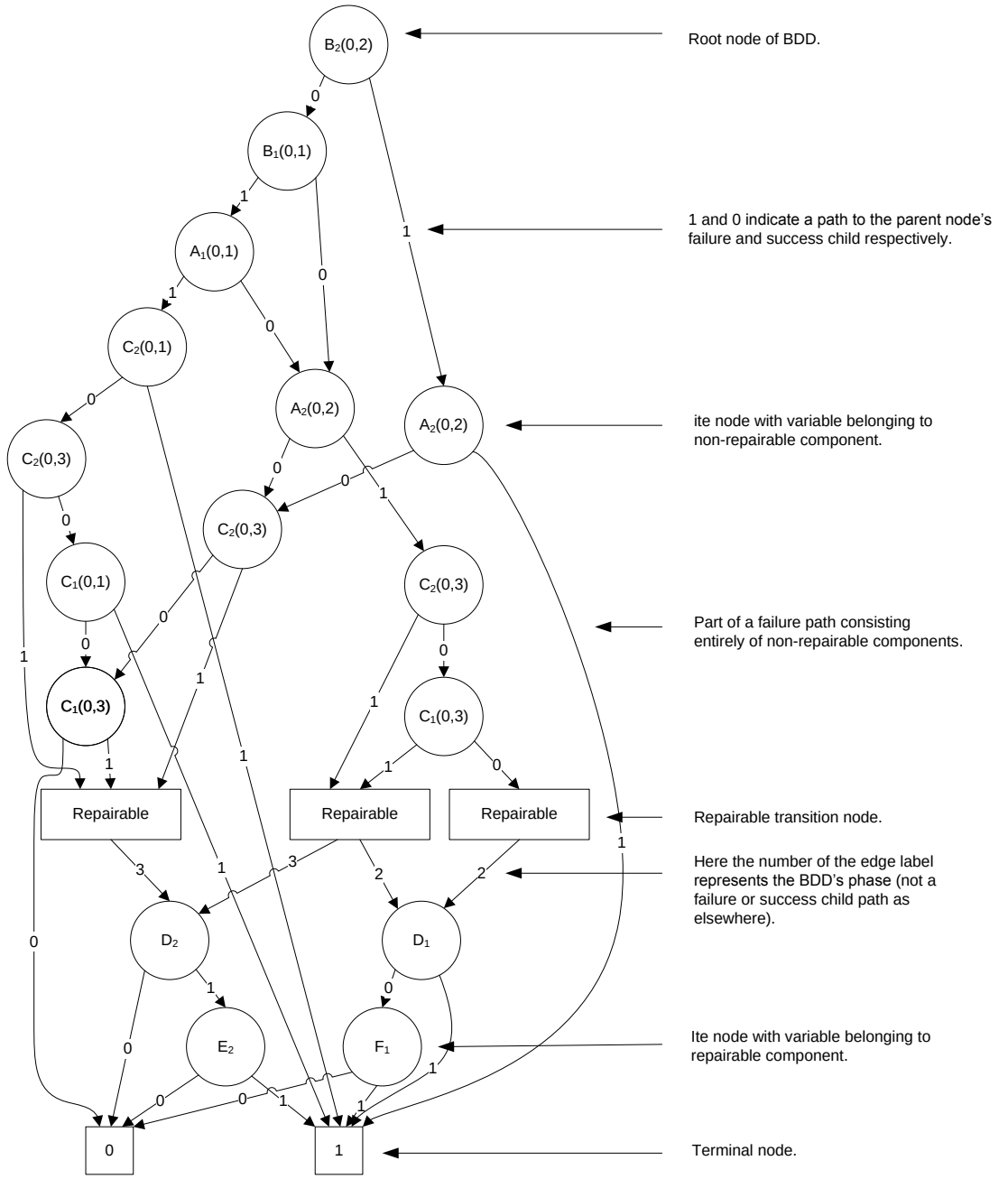


Figure 6.6 – BDD for phased mission fault tree from Figure 4.10 constructed through method given in section 6.4.2.1.

6.4.2.2 BDD Probability Evaluation

The BDD evaluation procedure used in this method gives an upper bound for the unreliability of the phased mission.

The first step in the probability evaluation is to evaluate the probability for each of the Repairable Transition nodes in the BDD. The probability of a Repairable Transition node is calculated through steps *a*, *b* and *c* given below:

a. A state transition rate matrix is formed, using the method described in section 6.2.1, for the components that appear in the variables of any of its child node BDDs.
b. For each of its child nodes, form a reduced state transition rate matrix for the phase indicated by the label of the connecting edge. This is formed from the full matrix obtained in step *a* and the child node BDD using the method described in section 6.2.2.
c. Evaluate the probability that the mission fails, given that one of the non-repairable event combinations represented by a path to the node occurs, through the method described in section 6.2.1. In that method, the reduced state transition rate matrices derived in step *b* are used for the phases with a phase edge out of the node, the full state transition rate matrix derived in step *a* are used for any phases that do not have a phase edge out of the node but are earlier than the last phase for which a phase edge does exist, and an initial state probability vector is used that assumes that all components are working at the start of the mission.

The terminal nodes and *ite* nodes with variables belonging to non-repairable components are then converted into an Implicant Tree data structure using the method described in section 4.2.2.2, with each Repairable Transition node that is a child of an Event Node in the Implicant Tree data structure given the probability calculated through the procedure outlined above. The probability evaluated for the top node of the Implicant Tree then gives the upper bound on the unreliability of the phased mission.

Due to the manner in which the evaluation procedure links the non-repairable and repairable component events through the Repairable Transition nodes, it essentially treats all failures of the non-repairable components in a phase as occurring at the start of that phase when determining the mission outcomes for the repairable component events. This is because part of the evaluation procedure uses methods optimised for non-repairable systems and the time at which a failure occurs during a phase does not affect the outcome of a mission in non-repairable systems – only the phase is important. Since earlier failure of a non-repairable component cannot decrease the probability of mission failure, but may increase it, this causes the method to give an upper bound on the mission unreliability. Due to the source of the estimation error, its size decreases with:

- Increasing numbers of phases
- Fewer phases fault trees containing both non-repairable and repairable components
- In phases fault trees containing both non-repairable and repairable components – lower proportions of failure modes containing both non-repairable and repairable components
- In phase failure modes containing both non-repairable and repairable components - lower proportions of non-repairable components
- Fewer non-repairable components with failure rates that increase with age

For most systems of the type and complexity found in the real world, the error will be small and since the estimate is conservative it may be used to show that a system's reliability is within a certain desired level. Three

phased missions containing both non-repairable and repairable components and whose component failure and repair rates are shown in Table 6.1 will now be used as examples. In the example phased mission shown in Figure 6.7, the method gives an estimated unreliability of 0.054 compared to the true value of 0.041 calculated through the method by Kim and Park, an overestimate of 32%. The source of the error can be seen in its BDD that is shown in Figure 6.8. In that BDD, the probability of system failure is the probability that component *B* is failed at some point during phase 1 and component *A* also fails in that phase (as shown by the path from the BDD root to terminal one node), whereas the exact probability is the probability that *B* is failed at some point in phase 1 after *A* has failed in that phase – i.e. the error is the probability that *B* is failed and repaired during phase 1 and *A* fails after its last repair. In the example phased missions shown in Figure 6.9 and Figure 6.10, the method gives the exact answers.

Table 6.1 - Failure and repair rates for the components in the phased mission shown in Figure 6.7, Figure 6.9 and Figure 6.10.

Component	**Failure Rate**	**Repair Rate**
A	0.001	0 (non-repairable)
B	0.01	0.01
C	0.01	0.01

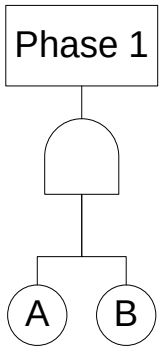


Figure 6.7 – Example phased mission for which method given mission unreliability upper bound.

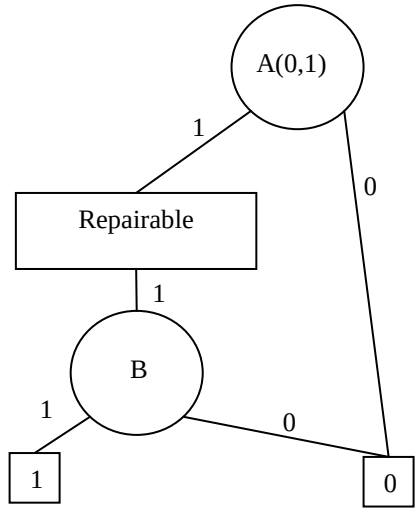


Figure 6.8 – BDD for phased mission fault tree in Figure 6.7.

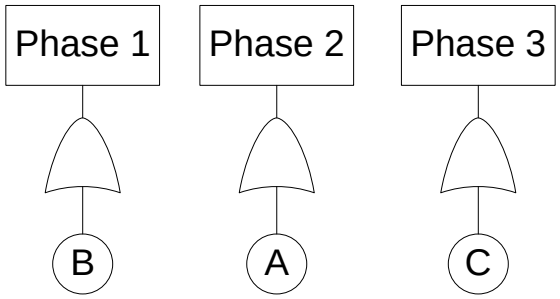


Figure 6.9 – Example phased mission for which method gives exact mission unreliability.

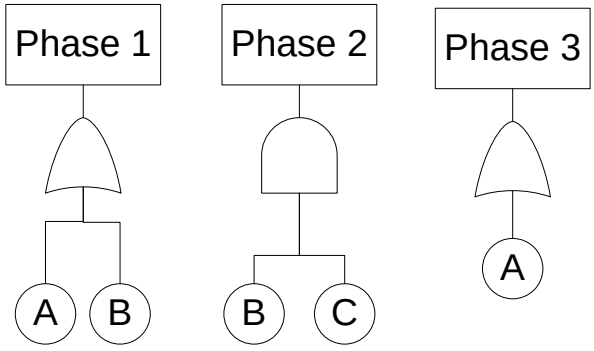


Figure 6.10 – Example phased mission for which method gives exact mission unreliability.

6.4.2.3 Software implementation and a worked example

A software tool has been developed that implements the method described above. The inputs to the tool are the phase fault trees and the failure and repair models for each component failure mode. This tool has been used to

analyse the example phased mission that is represented by the phased mission fault tree shown in Figure 4.10, where components *D*, *E* and *F* are repairable and each have a repair rate of 0.01 repairs per hour.

The BDD generated by the software is shown in Figure 6.6. It contains a total of 3 Repairable Transition nodes and its evaluation therefore requires the solution of 3 Markov models. Each of these Markov models is small, the largest consisting of three components (*D*, *F* and *E*) over two phases (2 and 3).

A worked example of the evaluation of the BDD paths that include the Repairable Transition node on the right in Figure 6.6, will now be described. As shown in Figure 6.6, the Repairable Transition node on the right describes the conditions in phase 2 for the two failure modes, D_1 and F_1, which determine whether the mission fails in that phase given the states of the non-repairable components indicated by the paths to that node. Following the procedure given in section 6.4.2.2, the first step in evaluating the probability for that node is to determine the full state transition rate matrix that includes components *D* and *F*, the components present in the variables of its child BDDs. The system states to be represented in that matrix are shown in Table 6.2, the Markov model showing the possible transitions between those states is shown in Figure 6.11 and the state indexes for the set of system states in each of the mission phases are given in Table 6.3.

Table 6.2 – System states for the repairable components from the BDD on the right in Figure 6.6.

	Failure Modes (1=Present, 0=Absent)			
State IDs	**D_1**	**D_2**	**F_1**	**F_2**
1	0	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	1	0	1	0
7	1	0	0	1
8	0	1	1	0
9	0	1	0	1

State Label	Present Failure Modes
1	None
2	D_1
3	D_2
4	F_1
5	F_2
6	D_1, F_1
7	D_1, F_2
8	D_2, F_1
9	D_2, F_2

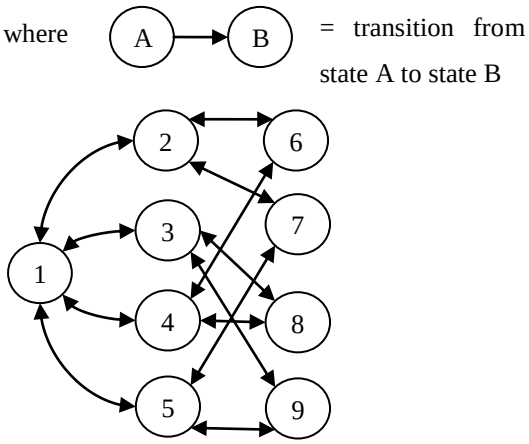


Figure 6.11 – Markov model for the repairable BDD on the right in Figure 6.6.

Table 6.3 – State indexes for each system state from Table 6.2 in the state transition matrices for phases 1 and 2 of the mission.

State IDs	Phase 1 State Index	Phase 2 State Index
1	1	1
2	2	System Failed State
3	3	2
4	4	System Failed State
5	5	3
6	6	System Failed State
7	7	System Failed State
8	8	System Failed State
9	9	4

At the start of the mission it is assumed that all components are in the working state, thus the probability of the system residing in state 1 from Table 6.2 at that time is 1 and the probability of the system residing in any other state is 0. This results in the state probability vector for the start of phase 1 shown in Equation 6.7, where the *i*th element represents the probability of being in the *i*th state.

$$P_1(0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \quad \textbf{6.7}$$

The Repairable Transition node has no edge labelled for phase 1 and therefore the full transition matrix is used to find the state probability vector at the end of phase 1. This matrix is shown in Equation 6.8, where element (*i*,*j*), $i \neq j$, represents the transition rate from state *j* to state *i*, and element (*i*,*i*) represents the transition rate out of the state *i* .

$$\beta_1 = \begin{bmatrix} -0.004 & 0.01 & 0.01 & 0.01 & 0.01 & 0 & 0 & 0 & 0 \\ 0.001 & -0.012 & 0 & 0 & 0 & 0.01 & 0.01 & 0 & 0 \\ 0.001 & 0 & -0.012 & 0 & 0 & 0 & 0 & 0.01 & 0.01 \\ 0.001 & 0 & 0 & -0.012 & 0 & 0.01 & 0 & 0.01 & 0 \\ 0.001 & 0 & 0 & 0 & -0.012 & 0 & 0.01 & 0 & 0.01 \\ 0 & 0.001 & 0 & 0.001 & 0 & -0.02 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0.001 & 0 & -0.02 & 0 & 0 \\ 0 & 0 & 0.001 & 0.001 & 0 & 0 & 0 & -0.02 & 0 \\ 0 & 0 & 0.001 & 0 & 0.001 & 0 & 0 & 0 & -0.02 \end{bmatrix} \quad \textbf{6.8}$$

Substituting the initial state probability vector and state transition rate matrix for phase 1 into Equation 5.4, along with the phase duration of 100 hours, gives the state probability vector for the end of phase 1 shown in Equation 6.9, where the *i*th element represents the probability of being in the *i*th state.

$$P_1(100) = [0.7806 \quad 0.05145 \quad 0.05145 \quad 0.05145 \quad 0.05145 \quad 0.003391 \quad 0.003391 \quad 0.003391 \quad 0.003391] \quad \textbf{6.9}$$

For phase 2, the BDD for the phase 2 edge out of the Repairable Transition node shows that the system fails if component *D* is failed in failure mode 1 or component *F* is failed in failure mode 1. Thus, only states 1, 3, 5 and 9 from Table 6.2 represent system working states in phase 2. Only the elements representing these working states from the end of the phase 1 state probability vector are used to form the initial state probability vector for phase 2, as shown in Equation 6.10, where the *i*th element represents the probability of being in the *i*th state.

$$P_2(0) = \begin{bmatrix} 0.7806 & 0.05145 & 0.05145 & 0.003391 \end{bmatrix}^T \quad \textbf{6.10}$$

The elements from the end of phase 1 state probability vector that represent system failure states in phase 2, i.e. states 2, 4, 6, 7 and 8, are summed to give the probability of failure on transition to phase 2 of 0.1131. The reduced state transition rate matrix is then formed by removing from the full state transition rate matrix,

Equation 6.8, all elements that represent transitions from, and to, phase 2 system failure states. The resultant reduced state transition rate matrix is shown in Equation 6.11 and the indexes for those states are given in Table 6.3.

$$\beta_2 = \begin{bmatrix} -0.004 & 0.01 & 0.01 & 0.0 \\ 0.001 & -0.012 & 0 & 0.01 \\ 0.001 & 0 & -0.012 & 0.01 \\ 0 & 0.001 & 0.001 & -0.02 \end{bmatrix} \quad \textbf{6.11}$$

Substituting the initial state probability vector and state transition rate matrix for phase 2 into Equation 5.4, along with the phase duration of 100 hours, gives the state probability vector for the end of phase 2 shown in Equation 6.12.

$$P_2(100) = \begin{bmatrix} 0.6164 & 0.05769 & 0.05769 & 0.005399 \end{bmatrix}^T \quad \textbf{6.12}$$

Since no further phases have edges out of the Repairable Transition node, deducting the sum of the elements in this state probability vector from 1 gives the probability value for this node as 0.2628.

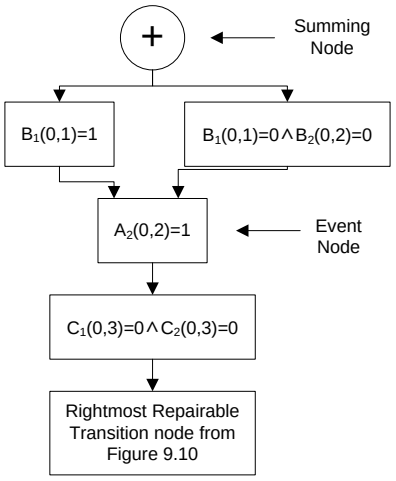


Figure 6.12 – Part of Implicant Tree data structure from BDD shown in Figure 6.6.

The BDD in Figure 6.6 shows that there are two possible paths from the top node to the Repairable Transition node shown on the right of Figure 6.6. The Implicant Tree data structure representation of these paths is shown in Figure 6.12. The probability of the Implicant Tree data structure if the probability for the Repairable Transition node were to have a value of 1, using the Implicant Tree evaluation procedure given in section 6.4.2.2, is 0.03244. Multiplying this with the probability of the Repairable Transition node that was calculated earlier gives the total contribution to mission unreliability from the Implicant Tree structure shown in Figure 6.12 as 0.00853.

6.4.2.4 Lower computation expense than alternatives

The exact mission unreliability for a repairable phased mission can be calculated through the method extended from Kim and Park given in section 6.2, assuming no computational errors are introduced in the evaluation of the matrix exponentials. The difference in computational expense between calculating the mission unreliability through that method and the upper bound value given by this new method, using the example mission to demonstrate, will now be discussed. The system in the example has 6 components, each with two failure modes, therefore the order of the full state transition rate matrix for the example mission, calculated through Equation 6.1, is 729. From this the three reduced state phase transition rate matrices are formed by removing the states that represent system failure. The largest of the resultant reduced state transition rate matrices, those for phases 1 and 3, are of order 675 since 54 of the states from the full matrix in each of these phases represent system failed states and can be removed. For example, in phase 1, the system failure states are all system states where component *A* is failed in failure mode 1, component *B* is failed in failure mode 1 and component *C* is failed in any failure mode. In comparison, the approximate solution requires the calculation of the exponential of 8 much smaller matrices, between order 4 and order 27. Since evaluating a matrix exponential is typically an *O(n^3)* calculation, evaluating the matrix exponential of 8 matrices where $n \leq 27$ is far more efficient than the solution of 3 matrices where $n \leq 675$. The greater number of states in the exact method also means greater computational expense in evaluating the Markov states representing system failure, e.g. using the method given in section 6.2.2.

The approximate method does however require the calculation of the values of the nodes in the non-repairable parts of the Implicant Tree, consisting of exponential cumulative distribution function calculations which are then multiplied and summed. However, the computational expense of this is negligible in comparison to the solution of the matrix exponentials.

6.4.2.5 Summary

A new method for the analysis of phased mission reliability of systems with both non-repairable and repairable multiple failure mode components, where repairable components are integrated back into the system immediately upon repair, has been developed. The method results in much smaller Markov state spaces and therefore faster analysis than when the method for systems with repairable components from Kim and Park is used. The mission unreliability value given is an estimate and not an exact value, although it is an upper bound and its accuracy increases with higher numbers of phases, corresponding to the type of phased mission for which the method gives the greatest improvement in computational efficiency over alternatives.

6.5 Summary and Conclusions

The literature review of methods for the analysis of repairable phased missions showed that whilst a large number of methods had been developed, each was constrained to the analysis of systems with particular features, such as where component repairs are integrated back into the system only at the end of phases. It also showed that they were, in comparison to the non-repairable system methods, limited to the analysis of small systems due to having much lower computational efficiency. This motivated research into the development of improvements to those methods for increased computational efficiency and applicability to a wider range of

systems. The absence of any method developed specifically for the analysis of systems containing both repairable and non-repairable components was also noted, despite such systems being prevalent in the real world. This led to research into the development of such methods with the aim of producing methods that combined the higher efficiency of the non-repairable system methods with the ability of the repairable methods to deal with the dependencies introduced by repair. The Markov modelling technique by Kim and Park was extended to the analysis of a phased mission system with multiple failure mode components and a BDD based method introduced that allows the evaluation of system state during the phase by phase state transition rate matrix reductions to be performed in an efficient manner. The improvements to the method by Kim and Park therefore extend the range of systems to which it can be applied and enable them to be analysed efficiently. An alternative method for the evaluation of the BDD in the method from Wang and Trivedi has also been developed that removes the inefficiencies present in their algorithm, offering a slight performance advantage particularly in applications such as real time analysis. Two new methods for the analysis of a phased mission system containing both repairable and non-repairable components have been introduced. Each of these is suited to the analysis of different system types, the first for systems where repairable components are integrated back into the system at the end of a phase and the second where they are integrated immediately and have multiple failure modes at the system level. Both methods offer a performance advantage over the methods for the analysis of repairable components. The significance of that advantage grows with increasing proportions of non-repairable components.

7 Literature Review of Reliability Importance Measures

7.1 Introduction

Importance measures are used to determine the contribution of each component in a system to a particular aspect of the overall reliability of the system. They can highlight the weak links in the system, show the potential for system reliability improvement and indicate the optimum strategy for improving the systems reliability. Typical tasks that utilise importance measures include finding the optimum component to improve in order to achieve a certain reduction in system risk and showing areas of the system where added redundancy would be most beneficial.

The majority of research into importance measures has focused on the measurement of the importance of components in non-phased missions. The most widely known and used importance measures are the risk achievement worth (RAW), the risk reduction worth (RRW), the Birnbaum importance measure and the Criticality importance measure. The differential importance measure (DIM) is a relatively new importance measure which has a number of useful properties. A number of importance measures focusing on measuring the importance of groups of components and the relationship between the importance of a component and the unreliability of another have also been developed. Another significant development is the recent introduction of conditional importance measures. With real time systems reliability analysis becoming more feasible, due to advances in available computing power and the introduction of new methods such as binary decision diagrams, the ability for importance measures to reflect the current status of a system is desirable. Recently, phased mission importance measures, based on the non-phased Birnbaum and Criticality importance measures, have been developed.

For each importance measure in this literature review, the defining equation is given, followed by demonstrative calculations for the components from an example system or phased mission. The importance measures discussed in this literature review are easily calculated from the definition given by their equation, provided that the reliability structure and component reliabilities are known. However, it can be difficult to visualise what each importance measure measures and defines as component importance from this alone. For this reason, the contributing event combinations for each component from an example system and their common properties are given for each importance measure. The characteristics of each importance measure, i.e. what each defines as component importance to system reliability, are determined by the types of event combinations that it assigns as contributing to the importance of each component. The relative importance given to each component by an importance measure depends on the sum of the probabilities from its contributing event combinations relative to those of the other components. The higher the likelihood of the event combinations that contribute to a certain component occur compared to those of another component, for a certain importance measure, the higher its relative importance. The absolute importance measure values for each component can be determined from a importance measure dependant function of this sum and some component independent value such as the system unreliability.

7.2 Non-Phased Missions

In this section, the existing research on importance measures developed for non-phased missions are discussed.

7.2.1 Example System

The system shown in Figure 7.1 will be used as an example to demonstrate and explain the importance measures discussed here.

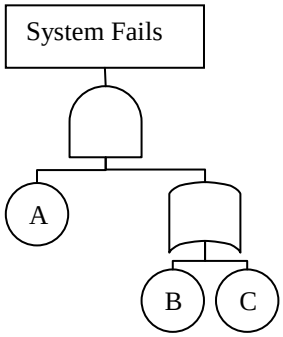


Figure 7.1 - Example System Fault Tree

The two cut sets for this system are shown in Equations 7.1 and 7.2.

$$C^1 = A \wedge B \qquad \textbf{7.1}$$

$$C^2 = A \wedge C \qquad \textbf{7.2}$$

The probabilities of component failure, for each component in the example system, are given in Table 7.1.

Table 7.1 - Example System Failure Probabilities.

Component Failure Event	**Probability**
A	0.01
B	0.02
C	0.01

Through the inclusion-exclusion expansion the system unreliability can be determined in terms of the component failure component probabilities as shown in Equation 7.3, where q_i represents the probability that component *i* fails.

$$Q_{SYS} = P((A \wedge B) \vee (A \wedge C)) = q_A q_B + q_A q_C - q_A q_B q_C \quad \textbf{7.3}$$

Using the failure event probabilities from Table 7.1 and Equation 7.3 the probability of system failure for the example system, Q_{SYS}, is calculated as shown below:

$$Q_{SYS} = q_A q_B + q_A q_C - q_A q_B q_C = 0.01 \times 0.02 + 0.01 \times 0.01 - 0.01 \times 0.02 \times 0.01 = 0.000298$$

The exhaustive mutually exclusive event combinations that can occur in the example system, along with their respective probabilities of occurrence, are shown in Table 7.2, where A is the failure event and $\overline{A}$ is the success event for component *A* for example.

For each importance measure discussed in section 7.2, the event combinations from Table 7.2 that contribute towards the importance of each component in the example system will be shown to help explain what each is measuring.

Table 7.2 – Probabilities for each mutually exclusive event combination in the example system.

Event combination	**Probability**
$\overline{A}\overline{B}\overline{C}$	0.960498
$A\overline{B}\overline{C}$	0.009702
$\overline{A}B\overline{C}$	0.019602
$\overline{A}\overline{B}C$	0.009702
$AB\overline{C}$	0.000198
$A\overline{B}C$	0.000098
$\overline{A}BC$	0.000198
ABC	0.000002

7.2.2 Single Event Importance Measures

7.2.2.1 Risk Achievement Worth

The Risk Achievement Worth (RAW) [54] measures the 'worth' of a component in achieving the current level of system reliability, i.e. how much better the system reliability is than it would be if the component's reliability

were at its lowest. It therefore indicates the importance of maintaining the component's reliability by showing the magnitude of the maximum potential deterioration in system reliability if it were not. The value of the RAW for a component signifies the ratio between the system unreliability if the component were to always fail and the current system unreliability, i.e. the multiple of the current system unreliability that would be seen if the component were replaced by a component that maximised system unreliability.

The RAW for component *i*, a_i, is defined as the system unreliability with component *i* failed, $Q_{SYS}(q_i = 1)$, normalised by the base system unreliability, Q_{SYS}, as shown in Equation 7.4. Its range is 1 to $\frac{1}{Q_{SYS}}$ for coherent systems, with a value of $\frac{1}{Q_{SYS}}$ signifying maximum importance. A RAW value of 5 for a component means that its current reliability maintains the system unreliability at a level that is 5 times lower than the level that would occur if it were replaced by a component whose reliability maximised system unreliability.

$$a_i = \frac{Q_{SYS}(q_i = 1)}{Q_{SYS}} \qquad \textbf{7.4}$$

The RAW for each component in the example system is calculated through Equation 7.4 as:

$$a_A = \frac{q_B + q_C - q_B q_C}{Q_{SYS}} = 100.000 \quad (1st)$$

$$a_B = \frac{q_A + q_A q_C - q_A q_C}{Q_{SYS}} = 33.557 \quad (= 2nd)$$

$$a_C = \frac{q_A q_B + q_A - q_A q_B}{Q_{SYS}} = 33.557 \quad (= 2nd)$$

Component *A* therefore has the highest worth in achieving the current level of system risk with a RAW value of 100 and it is therefore more important to maintain the reliability of this component in comparison to the others.Table 7.3 shows the failure combinations that contribute to the RAW for each component in the example system. It shows that event combinations that do not cause system failure but cause system failure when the component's event is replaced with its failure event, contribute to its RAW importance.

Table 7.3 – Event combinations that contribute to RAW for each component in the example system.

Event combination	Component A	Component B	Component C
$\bar{A}\bar{B}\bar{C}$	No	No	No
$A\bar{B}\bar{C}$	No	Yes	Yes
$\bar{A}B\bar{C}$	Yes	No	No
$\bar{A}\bar{B}C$	Yes	No	No
$AB\bar{C}$	No	No	No
$A\bar{B}C$	No	No	No
$\bar{A}BC$	Yes	No	No
ABC	No	No	No

The relationship between a_i and the sum of its contributing event combination probabilities, $P_{CEC_{a_i}}$, is shown in Equation 7.5.

$$a_i = \frac{Q_{SYS} + P_{CEC_{a_i}}}{Q_{SYS}} \qquad \textbf{7.5}$$

7.2.2.2 Risk Reduction Worth

The Risk Reduction Worth (RRW) [54] measures the level of improvement in system reliability that would be achieved if a component were made perfectly reliable. Ranking components with this measure indicates the component for which reliability improvement can potentially achieve the greatest improvement in the system's reliability.

The RRW for component *i*, r_i, is defined as the ratio of the base system unreliability, Q_{SYS}, to the system unreliability with component *i* working, $Q_{SYS}(q_i = 0)$, as shown in Equation 7.6. Its range is 1 to ∞ for coherent systems, with a value of ∞ signifying maximum importance. A RRW value of 2 for a component means that the current system unreliability is 2 times higher than it would be if it were replaced by a component whose reliability minimised the system unreliability.

$$r_i = \frac{Q_{SYS}}{Q_{SYS}(q_i = 0)} \qquad \textbf{7.6}$$

The RRW for the components in the example system are calculated through Equation 7.6 as:

$$r_A = \frac{Q_{SYS}}{0} = \infty \quad (1st)$$

$$r_B = \frac{Q_{SYS}}{q_A q_C} = 2.98 \quad (2nd)$$

$$r_C = \frac{Q_{SYS}}{q_A q_B} = 1.49 \quad (3rd)$$

The system is therefore perfectly reliable when component *A* is perfectly reliable, and an improvement in the reliability by a factor of almost 3 is achieved if component *B* were perfectly reliable.

Table 7.4 shows the event combinations that contribute to the RRW of each component in the example mission. It shows that event combinations that cause system failure but do not cause system failure when the component's event is replaced with its success event, contribute to the component's RRW.

Table 7.4 - Event combinations that contribute to RRW for each component in the example system.

Event combination	**Component A**	**Component B**	**Component C**
$\overline{A}\overline{B}\overline{C}$	No	No	No
$A\overline{B}\overline{C}$	No	No	No
$\overline{A}B\overline{C}$	No	No	No
$\overline{A}\overline{B}C$	No	No	No
$AB\overline{C}$	Yes	Yes	No
$A\overline{B}C$	Yes	No	Yes
$\overline{A}BC$	No	No	No
ABC	Yes	No	No

The relationship between r_i and the sum of its contributing event combination probabilities, $P_{CEC_{r_i}}$, is shown in Equation 7.7.

$$r_i = \frac{Q_{SYS}}{Q_{SYS} - P_{CEC_{r_i}}} \qquad \textbf{7.7}$$

Both RAW and RRW indicate system performance at the limits of component performance, due to this they are often less useful than the importance measures discussed later. Assuming the base system unreliability is known then the calculation of the RAW or RRW for a component requires a single additional system unreliability calculation and is therefore computationally inexpensive to calculate.

7.2.2.3 Birnbaum Importance Measure

The Birnbaum importance measure [55] for component *i*, IB_i, is defined as the partial derivative of the system failure probability, Q_{SYS}, with respect to the probability of component *i* failure, q_i, as shown in Equation 7.8. The value of the Birnbaum importance for a component signifies the ratio between the change in the unreliability of the component and the system, i.e. the gearing between them. Its range is 0 to 1 for coherent systems, with a value of 1 signifying maximum importance. A Birnbaum importance value of 0.6 for a component means that reducing its unreliability by a value of 0.1 leads to a 0.06 reduction in the system unreliability.

$$IB_i = \frac{\partial Q_{SYS}}{\partial q_i} \qquad \textbf{7.8}$$

It can be interpreted as a measure of the influence a change in the component unreliability has on system unreliability, with those with a high correlation ranked most important. Where it is possible to improve (increase) the reliability of any component by the same amount for the same cost, then the improvement should be applied to the component ranked highest by this measure.

A criticism of the Birnbaum importance measure is that it doesn't account for the probability of the failure of the component itself. If the base system unreliability is known, the calculation of Birnbaum importance measure requires two system unreliability calculations to be performed. The expense of a Birnbaum importance measure calculation is therefore approximately twice that of the RAW or RRW importance measures.

The Birnbaum importance for each component in the example system is calculated through Equation 7.8 as follows:

$$IB_A = q_B + q_C - q_B q_C = 0.0298 \quad (1st)$$

$$IB_B = q_A - q_A q_C = 0.0099 \quad (2nd)$$

$$IB_C = q_A - q_A q_B = 0.0098 \quad (3rd)$$

As shown, component *A* has the greatest BI importance whilst component *C* has the lowest.

Table 7.5 shows the event combinations that contribute to the Birnbaum importance of each component from the example system. It shows that combinations that cause system failure when the component's event is replaced with its failure event but not when replaced with its success event contribute to the component's Birnbaum importance, i.e. where the combination does not cause system failure without failure of the component, given the events for the other components.

Table 7.5 - Event combinations that contribute to Birnbaum importance for each component in the example system.

Event combination	Component A	Component B	Component C
$\overline{A}\overline{B}\overline{C}$	No	No	No
$A\overline{B}\overline{C}$	No	Yes	Yes
$\overline{A}B\overline{C}$	Yes	No	No
$\overline{A}\overline{B}C$	Yes	No	No
$AB\overline{C}$	Yes	Yes	No
$A\overline{B}C$	Yes	No	Yes
$\overline{A}BC$	Yes	No	No
ABC	Yes	No	No

The relationship between IB_i and the sum of its contributing event combination probabilities, $P_{CEC_{IB_i}}$, is shown in Equation 7.9.

$$IB_i = P_{CEC_{IB_i}} \qquad \textbf{7.9}$$

7.2.2.4 Criticality importance measure

The Criticality importance measure, which accounts for the component's probability of failure, is often cited as an improvement over the Birnbaum importance measure as it accounts for the component's reliability. The Criticality importance measures the influence of a component on the system unreliability and gives the proportion of the system failure probability caused by its failure probability. The components ranked highest by this importance measure are those for which a proportional reduction in failure probability results in the highest

reduction in system failure probability. Another advantage over the Birnbaum importance is that it is normalised by the overall system unreliability making comparisons between the importances of components from different systems more relevant.

The Criticality importance measure for component *i*, IC_i, is defined as the Birnbaum importance measure, IB_i, multiplied by the failure probability of component *i*, q_i, normalised by the system failure probability, Q_{SYS}, as shown in Equation 7.10. Its range is 0 to 1 for coherent systems, with a value of 1 signifying maximum importance. A Criticality value of 0.4 for a component means that replacing it with a perfectly reliable alternative would result in a 40% reduction in the probability of system failure.

$$IC_i = IB_i \cdot \frac{q_i}{Q_{SYS}} \qquad \textbf{7.10}$$

Where it is less expensive to improve the reliability of components with high unreliability by a given amount then the components ranked highest by this measure should be improved first. Dutuit and Rauzy [56] argue that components ranked higher by the Criticality importance should always be improved first, based on the reasoning that it is more difficult and costly to improve the more reliable components than to improve the less reliable ones. However this justification is not always valid. For example, a complex electronically controlled valve may be far less reliable than a pipe in a system, but it may still not be easier and cheaper to improve its reliability.

The Criticality importance for each component in the example system is calculated through Equation 7.10 as follows:

$$IC_A = \frac{IB_A \times q_A}{Q_{SYS}} = 1 \quad (1st)$$

$$IC_B = \frac{IB_B \times q_B}{Q_{SYS}} = 0.66 \quad (2nd)$$

$$IC_C = \frac{IB_C \times q_C}{Q_{SYS}} = 0.33 \quad (3rd)$$

This shows that component *A* is most important, with the greatest influence on system unreliability, whilst *C* is the least important with a third of the influence of *A*.

Table 7.6 shows the event combinations that contribute to the Criticality importance of each component from the example system. It shows that combinations that cause system failure but don't cause system failure when the component's event is replaced by its success event contribute to the component's Criticality importance, i.e. where the system fails due to the failure of the component but would not have otherwise, given the events for the other components.

Table 7.6 – Event combinations that contribute to Criticality importance for each component in the example system.

Event combination	Component A	Component B	Component C
$\overline{A}\overline{B}\overline{C}$	No	No	No
$A\overline{B}\overline{C}$	No	No	No
$\overline{A}B\overline{C}$	No	No	No
$\overline{A}\overline{B}C$	No	No	No
$AB\overline{C}$	Yes	Yes	No
$A\overline{B}C$	Yes	No	Yes
$\overline{A}BC$	No	No	No
ABC	Yes	No	No

The relationship between IC_i and the sum of its contributing event combination probabilities, $P_{CEC_{IC_i}}$, is shown in Equation 7.11.

$$IC_i = \frac{P_{CEC_{IC_i}}}{Q_{SYS}} \qquad \textbf{7.11}$$

7.2.2.5 Fussell-Vesely Importance Measure

The Fussell-Vesely importance measure differs from the Birnbaum and Criticality importance measures by utilising minimal cut sets to determine the contribution from component reliability to system reliability. Its value can be interpreted as the probability that a cut-set including the component has occurred given that the system is failed.

The Fussell-Vesely importance for component *i*, FV_i, is defined as the probability of the union of all minimal cut sets that contain component *i*, $P\left(\bigcup_{c\in C_i} c\right)$, where C_i is the set of minimal cut sets that contain component *i* ($C_i = \{x \in C : i \subset x\}$ where C is the set of all minimal cut sets), normalised by the system failure probability, Q_{SYS}, as shown in Equation 7.12. Its range is 0 to 1, with a value of 1 signifying maximum importance. A

Fussell-Vesely importance value of 0.2 for a component means that if the system has failed then there is a 0.2 probability that a cut-set containing that component has occurred.

$$FV_i = \frac{P\left(\bigcup_{c \in C_i} c\right)}{Q_{SYS}} \qquad \textbf{7.12}$$

The Fussell-Vesely importance for each component from the example system is calculated through Equation 7.12 as shown below:

$$FV_A = \frac{P((A \wedge B) \vee (A \wedge C))}{Q_{SYS}} = \frac{q_A q_B + q_A q_C - q_A q_B q_C}{Q_{SYS}} = 1 \quad (1st)$$

$$FV_B = \frac{P(A \wedge B)}{Q_{SYS}} = \frac{q_A q_B}{Q_{SYS}} = 0.671 \quad (2nd)$$

$$FV_C = \frac{P(A \wedge C)}{Q_{SYS}} = \frac{q_A q_C}{Q_{SYS}} = 0.336 \quad (3rd)$$

This shows that component *A* is the most important and component *C* the least important according to this importance measure. The Fussell-Vesely importance for component *B* shows that the probability of component *B* being failed given that the system is failed is 0.671.

Table 7.7 shows the event combinations that contribute to the Fussell-Vesely importance of each component from the example mission. It shows that event combinations that both cause system failure and contain the component's failure event contribute to the component's Fussell-Vesely importance. The relationship between FV_i and the sum of its contributing event combination probabilities, $P_{CEC_{FV_i}}$, is shown in Equation 7.13.

$$FV_i = \frac{P_{CEC_{FV_i}}}{Q_{SYS}} \qquad \textbf{7.13}$$

Table 7.7 - Event combinations that contribute to Fussell-Vesely importance for each component in the example system.

Event combination	Component A	Component B	Component C
$\overline{A}\overline{B}\overline{C}$	No	No	No
$A\overline{B}\overline{C}$	No	No	No
$\overline{A}B\overline{C}$	No	No	No
$\overline{A}\overline{B}C$	No	No	No
$AB\overline{C}$	Yes	Yes	No
$A\overline{B}C$	Yes	No	Yes
$\overline{A}BC$	No	No	No
ABC	Yes	Yes	Yes

7.2.2.6 Differential Importance Measure

The Differential importance measure (DIM) was developed by Borgonovo and Apostolakis [57] and gives the same importance rankings as either the Birnbaum or Criticality importance measure depending on the calculation method. A useful property of the DIM is that the importance of a group of components can be found by summing their individual importance values (see section 7.2.4).

The total variation of the system unreliability, ΔQ_{SYS}, due to a small variation in the unreliability of its components is expressed by the differential of the system unreliability function, as shown in Equation 7.14, where Δq_i is the variation in the unreliability of component *i* and *n* is the number of components in the system.

$$\Delta Q_{SYS} = \frac{\partial Q_{SYS}}{\partial q_1}\Delta q_1 + \frac{\partial Q_{SYS}}{\partial q_2}\Delta q_2 + \ldots + \frac{\partial Q_{SYS}}{\partial q_n}\Delta q_n \qquad \textbf{7.14}$$

The DIM for component *i*, DIM_i, is defined as the fraction of the total change in the system unreliability due to a change in the unreliability of component *i*, as shown in Equation 7.15. It has a range of 0 to 1 for coherent systems, with a value of 1 signifying maximum importance.

$$DIM_i = \frac{\frac{\partial Q_{SYS}}{\partial q_i}\Delta q_i}{\sum_{s=1}^{n}\frac{\partial Q_{SYS}}{\partial q_s}\Delta q_s} \qquad \textbf{7.15}$$

Equation 7.15 can be calculated with either a uniform or identical proportion variation in component unreliability. Where Δq_i is uniform for each component, DIM gives the same rankings as the Birnbaum importance. Where Δq_i represents an identical proportional variation to the original component reliability, DIM gives the same rankings as the Criticality importance. The DIM also enables the importance of groups of components to be calculated very easily, see section 7.2.4.1 for the Group DIM.

The DIM importance values for each component in the example system, calculated through Equation 7.15, and using uniform 0.01 changes to the component reliabilities, are as follows:

$$DIM_A = \frac{(q_B + q_C - q_B q_C)\times 0.01}{((q_B + q_C - q_B q_C)+(q_A - q_A q_C)+(q_A - q_A q_B))\times 0.01} = 0.602 \quad (1st)$$

$$DIM_B = \frac{(q_A - q_A q_C)\times 0.01}{((q_B + q_C - q_B q_C)+(q_A - q_A q_C)+(q_A - q_A q_B))\times 0.01} = 0.200 \quad (2nd)$$

$$DIM_C = \frac{(q_A - q_A q_B)\times 0.01}{((q_B + q_C - q_B q_C)+(q_A - q_A q_C)+(q_A - q_A q_B))\times 0.01} = 0.198 \quad (3rd)$$

This shows that component *A* has the highest importance by a significant margin, whereas component *B* has the second highest importance, closely followed by component *C*. The values are proportional to those given by the Birnbaum importance measure.

The DIM importance values for each component in the example system, calculated through Equation 7.15, and using uniform 1% changes to the component reliabilities, are as follows:

$$DIM_A = \frac{(q_B + q_C - q_B q_C)\times 0.01\times 0.01}{((q_B + q_C - q_B q_C)\times 0.01\times 0.01)+((q_A - q_A q_C)\times 0.02\times 0.01)+((q_A - q_A q_B)\times 0.01\times 0.01)}$$

$$= 0.502(1st)$$

$$DIM_B = \frac{(q_A - q_A q_C)\times 0.02\times 0.01}{((q_B + q_C - q_B q_C)\times 0.01\times 0.01)+((q_A - q_A q_C)\times 0.02\times 0.01)+((q_A - q_A q_B)\times 0.01\times 0.01)}$$

$$= 0.333(2nd)$$

$$DIM_C = \frac{(q_A - q_A q_B) \times 0.01 \times 0.01}{\left((q_B + q_C - q_B q_C) \times 0.01 \times 0.01\right) + \left((q_A - q_A q_C) \times 0.02 \times 0.01\right) + \left((q_A - q_A q_B) \times 0.01 \times 0.01\right)}$$

$$= 0.165(3rd)$$

This shows that component *A* has an importance approximately three times greater than component *C*, which itself has approximately half the importance of component *B* according to this importance measure. The values are proportional to those given by the Criticality importance measure.

The event combinations that contribute to the DIM importance, for uniform and proportional variations in component reliability, are the same as for the Birnbaum and Criticality importance, shown in Table 7.5 and Table 7.6 respectively.

7.2.3 Relationships between the importance measures

The relationship between the RAW, RRW, Birnbaums and Criticality importance measures, are shown in Table 7.8, where each row gives an importance measure in terms of the others (reproduced from Borgonovo [58]).

Table 7.8 - Relationships between importance measures.

Importance Measure	a_i **(RAW)**	r_i **(RRW)**	IB_i **(Birnbaum)**	I_{C_i} **(Criticality)**
a_i **(RAW)**	N/A	$\frac{IB_i}{Q_{SYS}} + r_i$ $\frac{IC_i}{q_i} + r_i$	$\frac{IB_i}{Q_{SYS}} + r_i$	$\frac{IC_i}{q_i} + r_i$
r_i **(RRW)**	$a_i - \frac{IC_i}{q_i}$	N/A	$a_i - \frac{IB_i}{Q_{SYS}}$	$a_i - \frac{IC_i}{q_i}$
IB_i **(Birnbaum)**	$Q_{SYS}(a_i - r_i)$	$Q_{SYS}(a_i - r_i)$	N/A	$IC_i \frac{Q_{SYS}}{q_i}$
IC_i **(Criticality)**	$q_i(a_i - r_i)$	$q_i(a_i - r_i)$	$IB_i \frac{q_i}{Q_{SYS}}$	N/A

7.2.4 First Order Group Importance Measures

The previous section discussed importance measures for individual components, but sometimes it is useful to determine the importance of groups of components instead. For example, the comparison of the relative importance of different subsystems, each containing numerous components, may be desired. The Group DIM, introduced below, is a first order group importance measure meaning that it does not account for the interactions

between components that cause the importance of components to change with the variation in the reliability of others. For this reason, it is best suited to measuring the importance of groups of components when only small variations in the reliability of the components are envisaged. Due to the possible presence of higher order interactions, it may not be optimal to improve the reliability of the components from a group that is ranked most important by a first order group importance measure if that improvement is large. In such cases, the higher order importance measures that are introduced later should be used instead.

7.2.4.1 Group DIM

The Group DIM, for the measurement of the importance of a group of components, was derived from the DIM by Borgonovo and Apostolakis [57]. One of the key advantages of DIM over importance measures such as Birnbaum importance is that it is additive for groups of components. This means that the DIM for a group of components is simply the sum of their individual DIM, calculated from Equation 7.15, as shown in Equation 7.16. The Group DIM for any combination of components therefore requires only minimal calculation once the individual component DIMs have been calculated.

$$DIM_{i,j,\ldots k} = DIM_i + DIM_j + \ldots + DIM_k \qquad \textbf{7.16}$$

The Group DIM for each component pair from the example system is calculated through Equation 7.16, using the individual DIM values for a uniform component reliability variation that were calculated in section 7.2.2.6, as follows:

$$DIM_{A,B} = DIM_A + DIM_B = 0.602 + 0.200 = 0.802 \quad (1st)$$

$$DIM_{A,C} = DIM_A + DIM_C = 0.602 + 0.198 = 0.800 \quad (2nd)$$

$$DIM_{B,C} = DIM_B + DIM_C = 0.200 + 0.198 = 0.398 \quad (3rd)$$

This shows that, according to the Group DIM, components *A* and *B* are the most important pair of components, with marginally higher importance than components *A* and *C*.

Although DIM indicates the importance of a group of components it can only be used to calculate the change in the system reliability for a small change in the group's component reliabilities. This is due to interaction terms that may be significant for larger changes and importance measures that address the impact of these interactions are discussed in the following section.

Table 7.9 shows the event combinations that contribute to the Group DIM for each pair of components from the example system, using uniform variation in component reliability. Where an event combination has a double contribution, this is indicated by '$\times 2$' in the table. The table shows that combinations that cause system failure when either component's event is replaced with its failure event but not when it is replaced with its success event contribute to the component's Birnbaum importance, i.e. where the system would not have failed unless the component failed, given the events for the other components. A double contribution occurs when this is the case for both components in the pair.

Table 7.9 - Event combinations that contribute to Group DIM importance for each component in the example system.

Event combination	Component Group A,B	Component Group A,C	Component Group B,C
$\overline{A}\overline{B}\overline{C}$	No	No	No
$A\overline{B}\overline{C}$	Yes	Yes	Yes ($\times 2$)
$\overline{A}B\overline{C}$	Yes	Yes	No
$\overline{A}\overline{B}C$	Yes	Yes	No
$AB\overline{C}$	Yes ($\times 2$)	Yes	Yes
$A\overline{B}C$	Yes	Yes ($\times 2$)	Yes
$\overline{A}BC$	Yes	Yes	No
ABC	Yes	Yes	No

7.2.5 Higher Order Group Importance Measures

The Group DIM does not take into account the higher order interactions between components that cause a change in the reliability of one component to result in changes to the importance of others. Where components appear in the same cut set, modifying the reliability of one component causes a change in the importance of other components from the same cut set. These dependencies therefore exist in any system which contains one or more cut sets that are greater than first order (i.e. contain more than a single component). Most real world systems have failure modes involving the failure of multiple components and therefore have these dependencies present.

There are many cases when it is necessary to investigate the affect on system reliability of modifying the reliability of multiple components. For example, increasing the reliability of one component will often require a sacrifice in the reliability of another due to budget constraints. For example with a fixed budget, replacing one component with a more expensive, more reliable alternative may mean using a cheaper, lower reliability alternative for another. The dependencies mean that, to achieve improvements in system reliability with maximum efficiency, it is not sufficient to rank the importance of components in a system using an individual component importance measure and then improve the reliability of each component in turn, beginning with the highest ranked and moving down the list. Changes to the reliability of a component may affect the order of importance of the other components from the same system.

Where the proposed reliability changes to components are small it may suffice to use the Group DIM since the higher order component interactions are also likely to be small and it has the advantage of being comparably inexpensive to calculate. Where larger component reliability changes are proposed it is better to use the higher order importance measures discussed in this section.

7.2.5.1 Joint Reliability Importance Measure

Hong and Lie [59] introduced the Joint Reliability Importance Measure (JRI) to measure the interaction between two components. The JRI for components *i* and *j*, $JRI_{i,j}$, is defined as the partial derivative of the Birnbaum importance for component *i* with respect to the unreliability of component *j*, as shown in Equation 7.17. Its range is -1 to 1. From Equation 7.17 it can be shown that when $I_{B_{i,j}} > 0$, the importance of component *i* increases as the reliability of component *j* decreases. Where $I_{B_{i,j}} < 0$, the importance of component *i* decreases as the reliability of component *j* decreases.

$$JRI_{i,j} = \frac{\partial^2 Q_{SYS}}{\partial q_i \partial q_j} = \frac{\partial IB_i}{\partial q_j} \qquad \textbf{7.17}$$

The JRI for each pair of components from the example system, calculated through Equation 7.17, are as follows:

$$JRI_{A,B} = \frac{\partial(q_B + q_C - q_B q_C)}{\partial q_B} = 1 - q_C = 0.99$$

$$JRI_{A,C} = \frac{\partial(q_B + q_C - q_B q_C)}{\partial q_C} = 1 - q_B = 0.98$$

$$JRI_{B,C} = \frac{\partial(q_A - q_A q_C)}{\partial q_C} = -q_A = -0.01$$

Two examples of information gained from these JRI values are that component *A* becomes more important as the reliability of *B* is decreased, whereas component *B* becomes less important as the reliability of *C* is decreased. Equation 7.18 from Zio and Podofillini [60] can be used to calculate the change to the system unreliability due to a change in the unreliability of any two components for which the Birnbaums and Joint Reliability importance measures are known.

$$\Delta Q_{SYS} = \sum_{i=1}^{n} IB_i \Delta q_i + \sum_{i=1}^{n} \sum_{j>i=1}^{n} IB_{i,j} \Delta q_i \Delta q_j \qquad \textbf{7.18}$$

Gao, Cui and Li [61] extended the JRI to multiple components. The definition for the JRI for three components *i, j* and *k*, $JRI_{i,j,k}$, is shown in Equation 7.19.

$$JRI_{i,j,k} = \frac{\partial^k Q_{SYS}}{\prod_i^k \partial q_i} \qquad \textbf{7.19}$$

7.2.5.2 Group RAW

The Group RAW [54] measures the 'worth' of a group of components in achieving the current level of system reliability, indicating the importance of maintaining the reliability of each component in the group. The Group RAW for components *i, j* and *k*, $a_{i,j,k}$, is defined as the system unreliability with components *i, j* and *k* failed, $Q_{SYS}(q_i = 1, q_j = 1, q_k = 1)$, normalised by the base system unreliability, Q_{SYS}, as shown in Equation 7.20. Its range is 1 to $\frac{1}{Q_{SYS}}$ for coherent systems, with a value of $\frac{1}{Q_{SYS}}$ signifying maximum importance. A Group RAW value of 4 for a component group means that their current reliability maintains the system unreliability at a level that is 4 times lower than the level that would occur if they were replaced by components whose reliability maximised the system unreliability.

$$a_{i,j,k} = \frac{Q_{SYS}(q_i = 1, q_j = 1, q_k = 1)}{Q_{SYS}} \qquad \textbf{7.20}$$

As shown by Cheok et al [54] the RAW for a group of components cannot be derived from the sum of the individual component RAW. Cheok et al also show that setting each of the group component's failure probabilities to unity in the cut sets, analogous to the method for the single component RAW, also gives an incorrect result. In this assessment Cheok is, however, making the unstated assumption that the cut sets have not been fully expanded through the inclusion-exclusion expansion, such as the common case where the rare event approximation is being used. If the fully inclusion-exclusion expanded cut sets are used then the method where each of the group's component failure probability is given the value of unity gives the correct result without any need for renaming and minimisation.

An example of the error in the Group RAW calculation when using the rare event approximation is shown in the incorrect calculation given below, where the group of components for which the Group RAW is calculated are X^1, X^2, X^3 and X^4:

$$Q_{SYS} = \left(A \wedge B \wedge (X^1 \vee X^3)\right) \vee \left(D \wedge E \wedge (X^3 \vee X^4)\right) \vee \left(F \wedge (X^1 \vee X^2) \wedge (X^2 \vee X^4)\right) \vee \left(G \wedge H\right)$$

$$a_{X^1,X^2,X^3,X^4} \neq \frac{2q_A q_B + 2q_D q_E + 4q_F + q_G q_H}{q_A q_B (q_{X^1} + q_{X^3}) + q_D q_E (q_{X^2} + q_{X^4}) + q_F (q_{X^1} + q_{X^3})(q_{X^2} + q_{X^4}) + q_G q_H}$$

It can be seen that the probability equation that results from the event unreliabilities being set to 1 is not correct, for example the probability of ABX^1 or ABX^2 occurring is represented by $2q_A q_B$, instead of the correct $q_A q_B$. Cheok shows that correct results may be obtained from the non fully expanded cut sets by first renaming each of

the group's components with the same identifier, re-minimising the resultant cut sets and then setting the group's components failure probability to unity. This is shown below:

$$Q_{SYS} = ABX \ + DEX + FC + GH$$

$$a_{C_1,C_2,C_3,C_4} = \frac{q_A q_B + q_D q_E + q_F + q_G q_H}{q_A q_B (q_{X_1} + q_{X_3}) + q_D q_E (q_{X_2} + q_{X_4}) + q_F (q_{X_1} + q_{X_3})(q_{X_2} + q_{X_4}) + q_G q_H}$$

Note that above result is an approximation as the higher order cut sets from the inclusion-exclusion expansion are ignored.

The Group RAW for each pair of components in the example system are calculated through Equation 7.20 as follows:

$$a_{A,B} = \frac{1 + q_C - q_c}{q_A q_B + q_A q_C - q_A q_B q_C} = 3356 \quad (=1st)$$

$$a_{A,C} = \frac{q_A + 1 - q_B}{q_A q_B + q_A q_C - q_A q_B q_C} = 3356 \quad (=1st)$$

$$a_{B,C} = \frac{q_A + q_A - q_A}{q_A q_B + q_A q_C - q_A q_B q_C} = 39 \quad (2nd)$$

As shown groups of components '*A,B'* and '*A,C'* have an extremely high Group RAW of 3356 since the failure of the components completes a minimal cut set, whereas the failure of '*B,C*' does not, increasing system failure probability by a far smaller factor of 39.

Table 7.10 shows the event combinations that contribute to the Group RAW for each pair of components from the example system. It shows that event combinations that do not cause system failure but cause system failure when the events for the group's components are all replaced with failure events, contribute to the group's Group RAW. The relationship between $a_{i,j,k}$ and the sum of its contributing event combination probabilities, $P_{CEC_{a_{i,j,k}}}$, is shown in Equation 7.21, i.e. the same form of relationship as the single component RAW, Equation 7.5.

$$a_{i,j,k} = \frac{Q_{SYS} + P_{CEC_{a_{i,j,k}}}}{Q_{SYS}} \qquad \textbf{7.21}$$

Table 7.10 - Event combinations that contribute to the Group RAW for each component in the example system.

Event combination	Component Group A,B	Component Group A,C	Component Group B,C
$\overline{A}\overline{B}\overline{C}$	Yes	Yes	No
$A\overline{B}\overline{C}$	Yes	Yes	Yes
$\overline{A}B\overline{C}$	Yes	Yes	No
$\overline{A}\overline{B}C$	Yes	Yes	No
$AB\overline{C}$	No	No	No
$A\overline{B}C$	No	No	No
$\overline{A}BC$	Yes	Yes	No
ABC	No	No	No

7.2.5.3 Group RRW

The Group RRW [54] measures the level of improvement in system reliability that would be achieved if all components in the group were made perfectly reliable. The Group RRW for components *i*, *j* and *k*, $r_{i,j,k}$, is defined as the ratio of the base system unreliability, Q_{SYS}, to the system unreliability with components *i*, *j* and *k* working, $Q_{SYS}(q_i = q_j = q_k = 0)$, as shown in Equation 7.22. This effectively removes all cut sets that contain any of the group's components from the numerator and hence re-minimisation is not required as it often is with the group RAW when using cut sets with the rare event approximation. Its range is 1 to ∞ for coherent systems, with a value of ∞ signifying maximum importance. A Group RRW value of 2 for a group of components means that the current system unreliability is 2 times higher than it would be if they were replaced by components whose reliability minimised the system unreliability.

$$r_{i,j,k} = \frac{Q_{SYS}}{Q_{SYS}(q_i = q_j = q_k = 0)} \qquad \textbf{7.22}$$

The Group RRW for each pair of components in the example system are calculated through Equation 7.22 as follows:

$$r_{A,B} = \frac{q_A q_B + q_A q_C - q_A q_B q_C}{0} = \infty \quad (=1st)$$

$$r_{A,C} = \frac{q_A q_B + q_A q_C - q_A q_B q_C}{0} = \infty \quad (=1st)$$

$$r_{B,C} = \frac{q_A q_B + q_A q_C - q_A q_B q_C}{0} = \infty \quad (=1st)$$

As shown, all component groups have the same, infinite, Group RAW since making both events in any group perfectly reliable results in perfect system reliability.

Table 7.11 shows the event combinations that contribute to the Group RRW for each pair of components from the example system. It shows that event combinations that cause system failure but do not cause system failure when the events for the group's components are all replaced with success events, contribute to the group's Group RRW. The relationship between $r_{i,j,k}$ and the sum of its contributing event combination probabilities, $P_{CEC_{r_{i,j,k}}}$, is shown in Equation 7.23, i.e. the same form of relationship as the single component RRW, Equation 7.7.

$$r_{i,j,k} = \frac{Q_{SYS}}{Q_{SYS} - P_{CEC_{r_{i,j,k}}}} \qquad \textbf{7.23}$$

Table 7.11 - Event combinations that contribute to Group RRW for each component in the example system.

Event combination	Component Group A,B	Component Group A,C	Component Group B,C
$\overline{A}\overline{B}\overline{C}$	No	No	No
$A\overline{B}\overline{C}$	No	No	No
$\overline{A}B\overline{C}$	No	No	No
$\overline{A}\overline{B}C$	No	No	No
$AB\overline{C}$	Yes	Yes	Yes
$A\overline{B}C$	Yes	Yes	Yes
$\overline{A}BC$	No	No	No
ABC	Yes	Yes	Yes

7.2.5.4 Differential Importance Measure II

Zio and Podofillini [60] introduced a second order differential importance measure known as the DIM^{II} that gives the relative influence that a variation in the reliability of a pair of components has on the system reliability compared to all component pairs in the system. The DIM^{II} for components *i* and *j*, $DIM_{i,j}^{II}$, is defined as the ratio of the change in the system unreliability due to a change in the reliability of components *i* and *j*, Δq_i and Δq_j respectively, to the sum of the changes in the system unreliability due to a change in the failure probability of each component pair in the system, where the component reliability variations are all of either a fixed proportion or of a uniform amount. This definition is shown in Equation 7.24, where $\Delta Q_{SYS_{i,j}}$ is defined by Equation 7.25.

$$DIM_{i,j}^{II} = \frac{\Delta Q_{SYS_{i,j}}}{\sum_{s=1}^{n}\sum_{t>s=1}^{n}\Delta Q_{SYS_{s,t}}} \qquad \textbf{7.24}$$

$$\Delta Q_{SYS_{i,j}} = \frac{\partial Q_{SYS}}{\partial q_i}\Delta q_i + \frac{\partial Q_{SYS}}{\partial q_j}\Delta q_j + \frac{\partial^2 Q_{SYS}}{\partial q_i \partial q_j}\Delta q_i \Delta q_j \qquad \textbf{7.25}$$

7.2.6 Conditional Importance Measures

Conditional importance measures can be used to determine the importance of components in a system when the state (failed or working) of other components are known. This can be useful in the real time reliability analysis of systems, where the reliability metrics reflect the current state of the system. If a certain system component changes from the working to failed state, the conditional importance measures can be used to determine the updated importance of the system's components. This has many practical applications, for example those components with increased importance, due to the failure of others, could be given additional protection from failure.

7.2.6.1 Conditional Birnbaum Importance Measure

The conditional Birnbaum importance [61] for component *i* for the case where component *j* is known to have failed, $IB_i(q_j = 1)$, is defined as the partial derivative of the system unreliability with component *j* failed, $Q_{SYS}(q_j = 1)$, with respect to the unreliability of component *i*. This definition is given by Equation 7.26, whilst the equivalent definition for the case where component *j* is known to have worked, $IB_i(q_j = 0)$, is given by Equation 7.27.

$$IB_i(q_j = 1) = \frac{\partial Q_{SYS}(q_j = 1)}{\partial q_i} \qquad \textbf{7.26}$$

$$IB_i(q_j = 0) = \frac{\partial Q_{SYS}(q_j = 0)}{\partial q_i} \qquad \textbf{7.27}$$

The calculations for the Birnbaum importance for component A and the conditional Birnbaum importance when component B is failed and working, for the example system, are calculated below:

$$IB_A = \frac{\partial Q_{SYS}}{\partial q_A} = q_B + q_C - q_B q_C = 0.0298$$

$$IB_A(q_B = 1) = \frac{\partial Q_{SYS}(q_B = 1)}{\partial q_A} = 1$$

$$IB_A(q_B = 0) = \frac{\partial Q_{SYS}(q_B = 0)}{\partial q_A} = q_C = 0.01$$

This shows that component *A* has the maximum importance possible when component *B* is known to fail and has an importance of almost one third of its non-conditional value when component *B* is known to work, according to the Birnbaum importance measure.

The relationship between JRI, Equation 7.17, and the conditional Birnbaum importance, Equation 7.26, is given by Equation 7.28.

$$IB_{i,j} = IB_i(q_j = 1) - IB_i(q_j = 0) \qquad \textbf{7.28}$$

The conditional importance measure shown is not limited to the case where the status of a single component is known, it may easily be extended to measure the importance of a component when the status of any number of components is known.

7.3 Phased Missions

In this section, the importance measures that were developed by Andrews [2] for phased missions are discussed. The importance measures presented here are suitable for the analysis of any phased mission system with coherent phase fault trees.

7.3.1 Example Missions

Two simple two phased missions, phased mission A and phased mission B, whose phase fault trees are shown in Figure 7.2a and Figure 7.2b respectively, will be used as an example to demonstrate the phased mission importance measures. These two example missions were chosen as, although very simple, together they contain all types of failures that occur in complex phased missions:

- In-phase mission failure caused by component failures in the same phase.
- In-phase mission failure caused by a combination of component failures in the same phase and earlier phases.
- Mission failure on transition to a phase caused by component failures in an earlier phase.

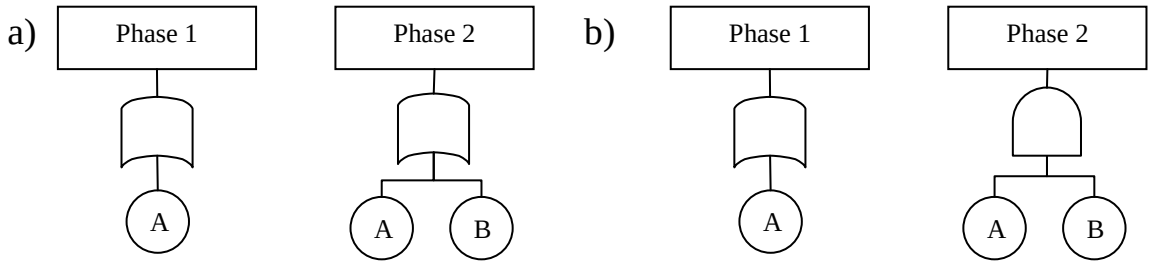


Figure 7.2 – Phase fault trees for example phased missions.

The probabilities of component failure in phases 1 and 2 are given in Table 7.12. The exhaustive mutually exclusive event combinations that can occur in the example missions, along with their respective probabilities of occurrence, are shown in Table 7.13, where $A(i, j)$ is the event that component *A* fails between the ends of phases *i* and *j* and $\overline{A}(0, j)$ is the event that component *A* does not fail before the end of phase *j*. For each importance measure discussed in section 7.3, the event combinations from Table 7.2 that contribute towards the importance of each component in each example mission will be shown to help explain what each is measuring.

Table 7.12 – Component phase failure probabilities.

Component Failure Event	Probability
$A(0,1)$	0.25
$A(1,2)$	0.50
$B(0,1)$	0.30
$B(1,2)$	0.20

Table 7.13 – Probabilities for each mutually exclusive event combination in the example mission.

Event combination	Probability
$\overline{A}(0,2)\overline{B}(0,2)$	0.125
$A(0,1)\overline{B}(0,2)$	0.125
$A(1,2)\overline{B}(0,2)$	0.250
$\overline{A}(0,2)B(0,1)$	0.075
$\overline{A}(0,2)B(1,2)$	0.050
$A(0,1)B(0,1)$	0.075
$A(0,1)B(1,2)$	0.050
$A(1,2)B(0,1)$	0.150
$A(1,2)B(1,2)$	0.100

7.3.1.1 Phased Mission A

Phased mission *A* will be analysed in this section. The phase one fault tree has the minimal cut sets {A} in phase 1 and {A}{B} in phase 2 as shown in Equations 7.29, 7.30 and 7.31, where C_j^i is the *ith* cut set in phase *j* and the event times given correspond to the end of phases.

$$C_1^1 = A(0,1) \quad \textbf{7.29}$$

$$C_2^1 = A(0,2) \quad \textbf{7.30}$$

$$C_2^2 = B(0,2) \quad \textbf{7.31}$$

The prime implicants for phase one failure, phase two failure, phase two transitional failure and mission failure are shown in Equations 7.32 to 7.35.

$$T_1 = A(0,1) \quad \textbf{7.32}$$

$$T_2 = A(1,2) \vee \left(\overline{A}(0,1) \wedge B(0,2)\right) \quad \textbf{7.33}$$

$$T_2^T = \overline{A}(0,1) \wedge B(0,1) \quad \textbf{7.34}$$

$$T_{Miss} = A(0,2) \vee B(0,2) \quad \textbf{7.35}$$

The equations for the probability of phase one, phase two, phase two transitional, phase two in-phase and mission failure are derived from Equations 7.33 to 7.34 using the inclusion-exclusion expansion, Equation 2.23, as shown in Equations 7.36 to 7.40. These equations are in the Henley and Inagaki form [62].

$$Q_1 = Q_1^P = P\left(A(0,1)\right) = q_{A_1} \quad \textbf{7.36}$$

$$Q_2 = P\left(A(1,2) \vee \left(\overline{A}(0,1) \wedge B(0,2)\right)\right) = q_{A_2} + p_{A_1} q_{B_{1,2}} - q_{A_2} q_{B_{1,2}} \quad \textbf{7.37}$$

$$Q_2^T = P\left(\overline{A}(0,1) \wedge B(0,1)\right) = p_{A_1} q_{B_1} \quad \textbf{7.38}$$

$$Q_2^P = q_{A_2} + p_{A_1} q_{B_{1,2}} - q_{A_2} q_{B_{1,2}} - p_{A_1} q_{B_1} = q_{A_2} + p_{A_1} q_{B_2} - q_{A_2} q_{B_{1,2}} \quad \textbf{7.39}$$

$$Q_{Miss} = P\left(A(0,2) \vee B(0,2)\right) = q_{A_{1,2}} + q_{B_{1,2}} - q_{A_{1,2}} q_{B_{1,2}} \quad \textbf{7.40}$$

Using the failure event probabilities from Table 7.12 together with Equations 7.36 to 7.40, the probability of failure in each period of the example mission are calculated as shown below:

$$Q_1 = Q_1^P = q_{A_1} = 0.25$$

$$Q_2 = q_{A_2} + p_{A_1} q_{B_{1,2}} - q_{A_2} q_{B_{1,2}} = 0.5 + (1 - 0.25) \times 0.5 - 0.5 \times 0.5 = 0.625$$

$Q_2^T = p_{A_1} q_{B_1} = (1 - 0.25) \times 0.3 = 0.225$

$Q_2^P = q_{A_2} + p_{A_1} q_{B_2} - q_{A_2} q_{B_{1,2}} = 0.5 + (1 - 0.25) \times 0.2 - 0.5 \times 0.5 = 0.400$

$Q_{Miss} = q_{A_{1,2}} + q_{B_{1,2}} - q_{A_{1,2}} q_{B_{1,2}} = 0.75 + 0.5 - 0.75 \times 0.5 = 0.875$

7.3.1.2 Phased Mission B

Phased mission *B* will now be analysed. The phase one fault tree has the minimal cut set shown in Equation 7.41, whilst the phase two fault tree has the minimal cut sets shown in 7.42, where C_j^i is the *ith* cut set in phase *j* and the event times given correspond to the end of phases.

$C_1^1 = A(0,1)$ **7.41**

$C_2^1 = A(1,2)B(0,2)$ **7.42**

The prime implicants for phase one failure, phase two failure, phase two transitional failure and mission failure are shown in Equations 7.43 to 7.46.

$T_1 = A(0,1)$ **7.43**

$T_2 = A(1,2) \wedge B(0,2)$ **7.44**

$T_2^T = 0$ **7.45**

$T_{Miss} = A(0,1) \vee \left(A(1,2) \wedge B(0,2)\right)$ **7.46**

The equations for the probability of phase one, phase two, phase two transitional, phase two in-phase and mission failure are derived from Equations 7.43 to 7.46 using the inclusion-exclusion expansion, Equation 2.23, as shown in Equations 7.47 to 7.50. Again, these equations are in the Henley and Inagaki form.

$Q_1 = Q_1^P = P\left(A(0,1)\right) = q_{A_1}$ **7.47**

$Q_2 = Q_2^P = P\left(A(1,2) \wedge B(0,2)\right) = q_{A_2} q_{B_{1,2}}$ **7.48**

$Q_2^T = 0$ **7.49**

$Q_{Miss} = P\left(A(0,1) \vee \left(A(1,2) \wedge B(0,2)\right)\right) = q_{A_1} + q_{A_2} q_{B_{1,2}}$ **7.50**

Using the failure event probabilities from Table 7.12 together with Equations 7.47 to 7.50, the probability of failure in each period of the example mission are calculated as shown below:

$$Q_1 = Q_1^P = q_{A_1} = 0.25$$

$$Q_2 = Q_2^P = q_{A_2} q_{B_{1,2}} = 0.5 \times 0.5 = 0.25$$

$$Q_2^T = 0$$

$$Q_{Miss} = q_{A_1} + q_{A_2} q_{B_{1,2}} = 0.25 + 0.5 \times 0.5 = 0.5$$

7.3.2 Component and system time periods in phased mission importance measures

Within a phased mission the effect of component failure on the system can vary significantly depending on the time at which the component fails whilst the consequence of system failure can vary significantly depending on the time at which the system fails. Precise knowledge of the component importance with regard to these time periods allows the correct decisions to be made on how to best improve the reliability of a phased mission. For this reason, a phased mission importance measure value for a component must specify the time periods for the component and system for which it was measured. Both the definitions for the phased mission importance measures from the literature review that are shown in this chapter and the new phased mission importance measures introduced in the next chapter refer to these time periods. This section therefore introduces some terminology and contains an in depth explanation of the significance of these time periods.

Each phased mission importance measure has a specified component failure period, which will be referred to as the ‘cause’ period. Each also has a specified system failure period, which will be referred to as the ‘effect’ period. The phased mission importance measures are measures of the contribution of failure of the component in the ‘cause’ period to system failure in the ‘effect’ period. Figure 7.3 gives an example where component failure in the ‘cause’ period consisting of the first half of phase 2 leads to an increased probability of system failure in the ‘effect’ period consisting of the second half of phase 3.

The ability to measure the importance of component failure from any time period to system failure in that or any other time period of the mission has many practical uses. For example, it allows the influence of component failure in each phase on the mission reliability to be compared, highlighting phases in which the components failure should be avoided. Similarly, measuring the importance of components to system failure in a specific period is useful in certain situations. Often system failure during a particular period (for example, the final two phases) may have greater consequence (for example, higher risk of fatalities) than at other times and therefore be of greater interest. In other cases, the failure of the system in a certain period may be of greater interest due to other external factors. For example, the probability of system failure in a particular period may be unacceptably high whilst being acceptable at other times (perhaps determined by industry regulations), thus the engineer who

wishes to improve mission performance in the unacceptable period will be interested in component importance to that period in particular.

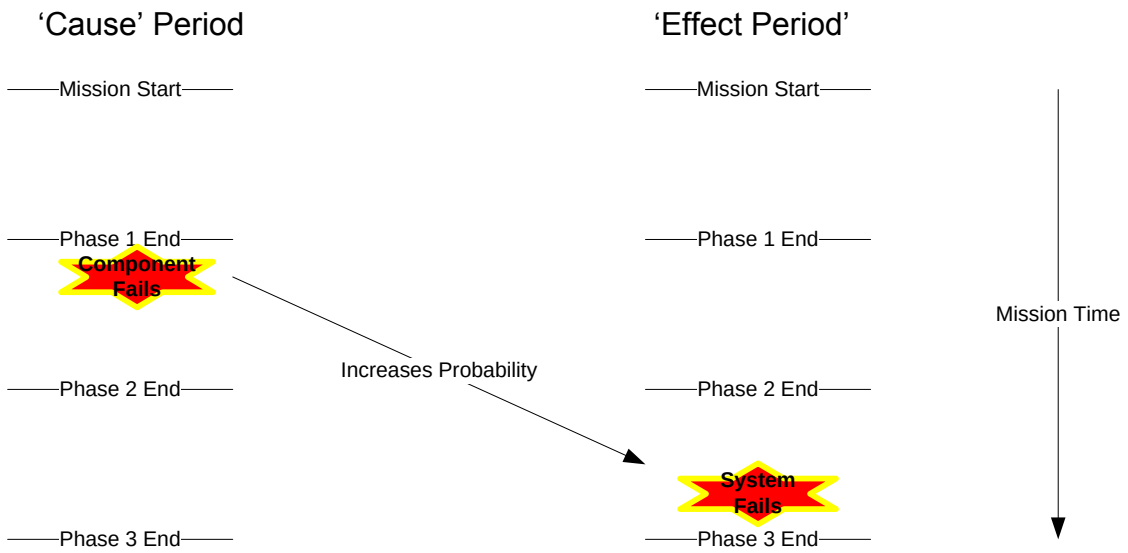


Figure 7.3 - Example of 'cause' and 'effect' periods for phased mission importance measures.

Some of the possible component and system failure period combinations that could be of interest to an analyst are the importance of the failure of a particular component in a particular phase to:

1. System failure in a particular phase.
2. System failure on transition to a particular phase.
3. In-phase system failure in a particular phase.
4. System failure during the complete phased mission.

and the overall contribution of the failure of a particular component to:

5. System failure in a particular phase.
6. System failure on transition to a particular phase.
7. In-phase system failure in a particular phase.
8. System failure during the complete phased mission.

7.3.3 Phased Birnbaum Importance Measure

7.3.3.1 In-Phase Birnbaum Importance Measure

Andrews [2] introduced the In-Phase Birnbaum importance measure for measuring the importance of a component to in-phase mission failure in a particular phase. The phase *j* in-phase 'effect' period Birnbaum importance measure for component *i*, $IB_{i_j}^{P}$, is defined as the partial derivative of the probability of in-phase

system failure in phase j, Q_j^P, with respect to the probability the component fails in phase j, q_{i_j}, as shown in Equation 7.51, where the derivative $\frac{\partial Q_j^P}{\partial q_{i_j}}$ must be evaluated in a specific way. Q_j^P must be formed as described by Henley and Inagaki [62] and then differentiated with the assumption that component success terms in Q_j^P are independent of q_{i_j}, i.e. $\frac{\partial p_{i_j}}{\partial q_{i_j}} = 0$. It has a range of 0 to 1, with a value of 1 signifying maximum importance.

$$IB_{i_j}^P = \frac{\partial Q_j^P}{\partial q_{i_j}} \qquad \textbf{7.51}$$

The phase 1 in-phase 'effect' period Birnbaum importance measure values for each component from example mission *A*, calculated through Equation 7.51, are shown below:

$$IB_{A_1}^P = \frac{\partial Q_1^P}{\partial q_{A_1}} = \frac{\partial q_{A_1}}{\partial q_{A_1}} = 1 \quad (1st)$$

$$IB_{B_1}^P = \frac{\partial Q_1^P}{\partial q_{B_1}} = \frac{\partial q_{A_1}}{\partial q_{B_1}} = 0 \quad (2nd)$$

This shows that component *A* has the maximum importance to phase 1 in-phase mission failure according to this importance measure, whilst component *B* has the minimum importance.

Table 7.14 shows the event combinations that contribute to the phase 1 in-phase 'effect' period Birnbaum importance measure value of each component in both example missions *A* and *B*, since the in-phase 1 mission failure conditions are the same for both missions.

The phase 2 in-phase 'effect' period Birnbaum importance measure values for each component from example mission *A*, calculated through Equation 7.51, are shown below:

$$IB_{A_2}^P = \frac{\partial Q_2^P}{\partial q_{A_2}} = \frac{\partial\left(q_{A_2} + p_{A_1} q_{B_2} - q_{A_2} q_{B_{1,2}}\right)}{\partial q_{A_2}} = 1 - q_{B_{1,2}} = 0.5 \quad (1st)$$

$$IB_{B_2}^P = \frac{\partial Q_2^P}{\partial q_{B_2}} = \frac{\partial\left(q_{A_2} + p_{A_1} q_{B_2} - q_{A_2} q_{B_{1,2}}\right)}{\partial q_{B_2}} = p_{A_1} - q_{A_2} = 0.25 \quad (2nd)$$

This shows that component *A* is most important to in-phase 2 mission failure according to this importance measure and has twice the importance of component *B*.

Table 7.14 – Event combinations that contribute to the phase 1 in-phase 'effect' period Birnbaum importance measure value of each component in example missions *A* and *B*.

Event combination	Component A	Component B
$\overline{A}(0,2)\overline{B}(0,2)$	Yes	No
$A(0,1)\overline{B}(0,2)$	Yes	No
$A(1,2)\overline{B}(0,2)$	Yes	No
$\overline{A}(0,2)B(0,1)$	Yes	No
$\overline{A}(0,2)B(1,2)$	Yes	No
$A(0,1)B(0,1)$	Yes	No
$A(0,1)B(1,2)$	Yes	No
$A(1,2)B(0,1)$	Yes	No
$A(1,2)B(1,2)$	Yes	No

Table 7.15 and Table 7.16 show the event combinations that contribute to the phase 2 in-phase 'effect' period Birnbaum importance measure value of each component in example mission *A* and *B* respectively. Table 7.14, Table 7.15 and Table 7.16 show that the event combinations that contribute to a component's phase *j* in-phase 'effect' period Birnbaum importance measure value are those that cause phase *j* in-phase system failure when the component's event is replaced by its phase *j* failure event but not when replaced by its phase *j* success event, i.e. where the events for the other components are such that phase *j* in-phase system failure would occur if the component failed in that phase but not if it were successful through the mission.

Table 7.15 - Event combinations that contribute to the in-phase 'effect' period Birnbaum importance for each component in example mission *A*.

Event combination	Component A	Component B
$\overline{A}(0,2)\overline{B}(0,2)$	Yes	Yes
$A(0,1)\overline{B}(0,2)$	Yes	No
$A(1,2)\overline{B}(0,2)$	Yes	No
$\overline{A}(0,2)B(0,1)$	No	Yes
$\overline{A}(0,2)B(1,2)$	No	Yes
$A(0,1)B(0,1)$	No	No
$A(0,1)B(1,2)$	No	No
$A(1,2)B(0,1)$	No	No
$A(1,2)B(1,2)$	No	No

The relationship between $IB_{i_j}^{P}$ and the sum of its contributing event combination probabilities, $P_{CEC_{IB_{i_j}^{P}}}$, is shown in Equation 7.52.

$$IB_{i_j}^{P} = P_{CEC_{IB_{i_j}^{P}}} \qquad \textbf{7.52}$$

Table 7.16 - Event combinations that contribute to the in-phase 'effect' period Birnbaum importance for each component in example mission *B*.

Event combination	Component A	Component B
$\overline{A}(0,2)\overline{B}(0,2)$	No	No
$A(0,1)\overline{B}(0,2)$	No	No
$A(1,2)\overline{B}(0,2)$	No	Yes
$\overline{A}(0,2)B(0,1)$	Yes	No
$\overline{A}(0,2)B(1,2)$	Yes	No
$A(0,1)B(0,1)$	Yes	No
$A(0,1)B(1,2)$	Yes	No
$A(1,2)B(0,1)$	Yes	Yes
$A(1,2)B(1,2)$	Yes	Yes

7.3.3.2 Transitional Birnbaum Importance Measure

Andrews [2] also presented the Transitional Birnbaum importance measure for measuring the importance of a component to the probability of mission failure on transition to a particular phase. The phase j transition 'effect' period Birnbaum importance measure with a phase k 'cause' period for component i, $IB^T_{i_{j,k}}$, is defined as the partial derivative of the probability of system failure on transition to phase j, Q^T_j, with respect to the probability the component fails in phase k, q_{i_k}, as shown in Equation 7.53, where Q^T_j is in the Henley and Inagaki form and it is assumed that $\frac{\partial p_{i_k}}{\partial q_{i_k}} = 0$. It has a range of 0 to 1, with a value of 1 signifying maximum importance.

$$IB^T_{i_{j,k}} = \frac{\partial Q^T_j}{\partial q_{i_k}} \qquad \textbf{7.53}$$

The calculations of the phase 2 transition 'effect' period Birnbaum importance measure with phase 1 'cause' period values are given below for each component from example mission *A*:

$$IB^{T}_{A_{2,1}} = \frac{\partial Q^{T}_{2}}{\partial q_{A_1}} = \frac{\partial\left(p_{A_1} q_{B_1}\right)}{\partial q_{A_1}} = 0 \quad (2nd), \text{ where } \frac{\partial p_{A_1}}{\partial q_{A_1}} \text{ is assumed to be 0.}$$

$$IB^{T}_{B_{2,1}} = \frac{\partial Q^{T}_{2}}{\partial q_{B_1}} = \frac{\partial\left(p_{A_1} q_{B_1}\right)}{\partial q_{B_1}} = p_{A_1} = 0.25 \quad (1st)$$

This shows that the failure of component *A* in phase 1 has no importance to system failure on transition to phase 2, whilst component *B* has the highest importance, according to this importance measure.

Table 7.17 - Event combinations that contribute to the value of the phase 2 transition 'effect' period Birnbaum importance measure with a phase 1 'cause' period for each component in example mission *A*.

Event combination	Component A	Component B
$\overline{A}(0,2)\overline{B}(0,2)$	No	Yes
$A(0,1)\overline{B}(0,2)$	No	No
$A(1,2)\overline{B}(0,2)$	No	Yes
$\overline{A}(0,2)B(0,1)$	No	Yes
$\overline{A}(0,2)B(1,2)$	No	Yes
$A(0,1)B(0,1)$	No	No
$A(0,1)B(1,2)$	No	No
$A(1,2)B(0,1)$	No	Yes
$A(1,2)B(1,2)$	No	Yes

Table 7.17 and Table 7.18 show the event combinations that contribute to the value of the phase 2 transition 'effect' period Birnbaum importance measure with phase 1 'cause' period for each component from example mission *A* and *B* respectively. They show that the event combinations that contribute to a component's phase *j* transition 'effect' period Birnbaum importance measure with a phase *k* 'cause' period are those that cause system failure on transition to phase *j* when the component's event is replaced by its phase *k* failure event, but not when replaced by its mission success event, i.e. where the events for the other components are such that

system failure on transition to phase *j* would occur if the component failed in phase *k* but not if it were successful through the mission.

Table 7.18 - Event combinations that contribute to the value of the phase 2 transition 'effect' period Birnbaum importance with a phase 1 'cause' period for each component in example mission *B*.

Event combination	Component A	Component B
$\overline{A}(0,2)\overline{B}(0,2)$	No	No
$A(0,1)\overline{B}(0,2)$	No	No
$A(1,2)\overline{B}(0,2)$	No	No
$\overline{A}(0,2)B(0,1)$	No	No
$\overline{A}(0,2)B(1,2)$	No	No
$A(0,1)B(0,1)$	No	No
$A(0,1)B(1,2)$	No	No
$A(1,2)B(0,1)$	No	No
$A(1,2)B(1,2)$	No	No

The relationship between $IB^{T}_{i_{j,k}}$ and the sum of its contributing event combination probabilities, $P_{CEC_{IB^{T}_{i_{j,k}}}}$, is shown in Equation 7.54.

$$IB^{T}_{i_{j,k}} = P_{CEC_{IB^{T}_{i_{j,k}}}} \qquad \textbf{7.54}$$

7.3.4 Phased Criticality Importance Measure

7.3.4.1 In-Phase Criticality Importance Measure

Andrews [2] presented the In-Phase Criticality importance measure to measure the importance of a component to the probability of in-phase mission failure in a particular phase. Unlike the In-Phase Birnbaum importance, Equation 7.51, this importance measure accounts for the reliability of the component and is normalised. The in-phase phase *j* 'effect' period Criticality importance measure for component *i*, $IC^{P}_{i_j}$, is defined as its phase *j* In-

Phase Birnbaum importance, $IB_{i_j}^P$, multiplied by its probability of failure in phase *j*, q_{i_j}, and normalised by the probability of system failure in phase *j*, Q_j, as shown in Equation 7.55. It has a range of 0 to $\frac{Q_j^P}{Q_j}$, with a value of $\frac{Q_j^P}{Q_j}$ signifying maximum importance (therefore being a maximum of 1 if all failures in phase *j* are in-phase failures).

$$IC_{i_j}^P = \frac{IB_{i_j}^P q_{i_j}}{Q_j} \qquad \textbf{7.55}$$

The values for the in-phase phase 1 'effect' period Criticality importance measure for each component from example mission *A*, calculated through Equation 7.55, are shown below:

$$IC_{A_1}^P = \frac{IB_{A_1}^P q_{A_1}}{Q_1} = \frac{1 \times 0.25}{0.25} = 1 \quad (1st)$$

$$IC_{B_1}^P = \frac{IB_{B_1}^P q_{B_1}}{Q_1} = \frac{0 \times 0.30}{0.25} = 0 \quad (2nd)$$

Component *A* therefore has the maximum importance to in-phase 1 mission failure according to this importance measure, whilst component *B* has the minimum importance.

Table 7.19 shows the event combinations that contribute to the in-phase phase 1 'effect' period Criticality importance measure value for each component in both example missions *A* and *B*, since the in-phase 1 mission failure conditions are the same for both missions.

Table 7.19 - Event combinations that contribute to the in-phase phase 1 'effect' period Criticality importance measure value for each component in example missions *A* and *B*.

Event combination	Component A	Component B
$\overline{A}(0,2)\overline{B}(0,2)$	No	No
$A(0,1)\overline{B}(0,2)$	Yes	No
$A(1,2)\overline{B}(0,2)$	No	No
$\overline{A}(0,2)B(0,1)$	No	No
$\overline{A}(0,2)B(1,2)$	No	No
$A(0,1)B(0,1)$	Yes	No
$A(0,1)B(1,2)$	Yes	No
$A(1,2)B(0,1)$	No	No
$A(1,2)B(1,2)$	No	No

The in-phase phase 2 'effect' period Criticality importance measure values for each component in example mission *A*, calculated through Equation 7.55, are shown below:

$$IC_{A_2}^{P} = \frac{IB_{A_2}^{P} q_{A_2}}{Q_2} = \frac{0.5 \times 0.50}{0.625} = 0.4 \quad (1st)$$

$$IC_{B_2}^{P} = \frac{IB_{B_2}^{P} q_{B_2}}{Q_2} = \frac{0.25 \times 0.20}{0.625} = 0.08 \quad (2nd)$$

This shows that component *A* has 5 times greater importance than component *B* to in-phase failure in phase 2 of the example mission.

Table 7.20 and Table 7.21 show the event combinations that contribute to the value of the in-phase phase 2 'effect' period Criticality importance measure for each component in example mission *A* and *B* respectively.

Table 7.20 - Event combinations that contribute to the in-phase phase 2 'effect' period Criticality importance measure value of each component in example mission *A*.

Event combination	Component A	Component B
$\overline{A}(0,2)\overline{B}(0,2)$	No	No
$A(0,1)\overline{B}(0,2)$	No	No
$A(1,2)\overline{B}(0,2)$	Yes	No
$\overline{A}(0,2)B(0,1)$	No	No
$\overline{A}(0,2)B(1,2)$	No	Yes
$A(0,1)B(0,1)$	No	No
$A(0,1)B(1,2)$	No	No
$A(1,2)B(0,1)$	No	No
$A(1,2)B(1,2)$	No	No

Together, Table 7.19, Table 7.20 and Table 7.21 show that an event combination contributes to a component's in-phase phase *j* 'effect' period Criticality importance if it includes the phase *j* component failure event and causes phase *j* in-phase system failure but not if the component's event were replaced with its mission success event; i.e. where the component's phase *j* failure event is present and the events for the other components are such that this causes phase *j* in-phase system failure that would not occur if the component were instead successful through the mission.

The relationship between $IC_{i_j}^P$ and the sum of its contributing event combination probabilities, $P_{CEC_{IC_{i_j}^P}}$, is shown in Equation 7.56.

$$IC_{i_j}^P = \frac{P_{CEC_{IC_{i_j}^P}}}{Q_j} \qquad \textbf{7.56}$$

Table 7.21 - Event combinations that contribute to the in-phase phase 2 'effect' period Criticality importance measure value of each component in example mission *B*.

Event combination	Component A	Component B
$\overline{A}(0,2)\overline{B}(0,2)$	No	No
$A(0,1)\overline{B}(0,2)$	No	No
$A(1,2)\overline{B}(0,2)$	No	No
$\overline{A}(0,2)B(0,1)$	No	No
$\overline{A}(0,2)B(1,2)$	No	No
$A(0,1)B(0,1)$	No	No
$A(0,1)B(1,2)$	No	No
$A(1,2)B(0,1)$	Yes	No
$A(1,2)B(1,2)$	Yes	Yes

7.3.4.2 Transitional Criticality Importance Measure

Andrews [2] introduced the Transitional Criticality importance measure to measure the importance of a component to the probability of mission failure on transition to a particular phase. Unlike the In-Phase Birnbaum importance, Equation 7.53, this importance measure accounts for the reliability of the component, includes its influence from all earlier phases and is normalised. The phase *j* transition 'effect' period Criticality importance measure for component *i*, $IC_{i_j}^T$, is defined as the sum of the products of the phase *j* transition 'effect' period Birnbaum importance measures with phase *k* 'cause' period, $IB_{i_{j,k}}^T$, and the probability the component fails in phase *k*, q_{i_k}, for all values of *k*, normalised by the probability of system failure in phase *j*, Q_j, as shown in Equation 7.57. It has a range of 0 to $\frac{Q_j^T}{Q_j}$, with a value of $\frac{Q_j^T}{Q_j}$ signifying maximum importance (therefore being a maximum of 1 if all failures in phase *j* are transitional failures).

$$IC_{i_j}^{T} = \frac{\sum_{1}^{j-1} IB_{i_{j,k}}^{T} q_{i_k}}{Q_j} \qquad \textbf{7.57}$$

The values of the phase 2 transition 'effect' period Criticality importance values for each component in example mission *A*, calculated through Equation 7.57, are shown below:

$$IC_{A_2}^{T} = \frac{\sum_{k=1}^{1} IB_{A_{2,k}}^{T} q_{A_k}}{Q_2} = \frac{0 \times 0.25}{0.625} = 0 \quad (2nd)$$

$$IC_{B_2}^{T} = \frac{\sum_{k=1}^{1} IB_{B_{2,k}}^{T} q_{B_k}}{Q_2} = \frac{0.25 \times 0.2}{0.625} = 0.08 \quad (1st)$$

This shows that component *B* has the highest importance for failure on transition to phase 2, with component *A* having zero importance.

Table 7.22 and Table 7.23 show the event combinations that contribute to the value of the phase 2 transition 'effect' period Criticality importance measure for each component in example mission *A* and *B* respectively. They show that an event combination contributes to a component's phase *j* transition 'effect' period Criticality importance measure value if it includes a failure event for the component from an earlier phase and causes system failure on transition to phase *j* but not if the component's event were replaced with its mission success event; i.e. where the component's failure event for an earlier phase is present and the events for the other components are such that this causes system failure to occur on transition to phase *j* that would not occur if the component were instead successful through the mission.

Table 7.22 - Event combinations that contribute to the phase 2 transition 'effect' period Criticality importance measure value of each component in example mission *A*.

Event combination	Component A	Component B
$\overline{A}(0,2)\overline{B}(0,2)$	No	No
$A(0,1)\overline{B}(0,2)$	No	No
$A(1,2)\overline{B}(0,2)$	No	No
$\overline{A}(0,2)B(0,1)$	No	Yes
$\overline{A}(0,2)B(1,2)$	No	No
$A(0,1)B(0,1)$	No	No
$A(0,1)B(1,2)$	No	No
$A(1,2)B(0,1)$	No	Yes
$A(1,2)B(1,2)$	No	No

The relationship between $IC_{i_j}^T$ and the sum of its contributing event combination probabilities, $P_{CEC_{IC_{i_j}^T}}$, is shown in Equation 7.58.

$$IC_{i_j}^T = \frac{P_{CEC_{IC_{i_j}^T}}}{Q_j} \qquad \textbf{7.58}$$

Table 7.23 - Event combinations that contribute to the phase 2 transition 'effect' period Criticality importance measure value of each component in example mission *B*.

Event combination	Component A	Component B
$\overline{A}(0,2)\overline{B}(0,2)$	No	No
$A(0,1)\overline{B}(0,2)$	No	No
$A(1,2)\overline{B}(0,2)$	No	No
$\overline{A}(0,2)B(0,1)$	No	No
$\overline{A}(0,2)B(1,2)$	No	No
$A(0,1)B(0,1)$	No	No
$A(0,1)B(1,2)$	No	No
$A(1,2)B(0,1)$	No	No
$A(1,2)B(1,2)$	No	No

7.3.4.3 Mission Criticality Importance Measure

Andrews [2] also presented an importance measure to measure the importance of a component to a complete phased mission. The Phased Mission Criticality importance measure for component *i*, I_i^M, is defined as the sum of the non-normalised In-Phase and Transitional Criticality importance measures (whose normalised definitions are given in Equations 7.55 and 7.57 respectively) for component *i* in each of the *m* phases of the mission, normalised by the probability of system failure over the complete mission, Q_{Miss}. This definition is shown in Equation 7.59. It has a range of 0 to 1, with a value of 1 signifying maximum importance.

$$IC_i^M = \frac{\sum_{j=1}^{m}\left\{\frac{\partial Q_j^P}{\partial q_{i_j}} q_{i_j} + \left(\sum_{k=1}^{j-1}\frac{\partial Q_j^T}{\partial q_{i_k}} q_{i_k}\right)\right\}}{Q_{Miss}} \qquad \textbf{7.59}$$

The Mission Criticality importance measure values for each component in example mission *A*, calculated through Equation 7.59, are shown below:

$$IC_A^M = \frac{\sum_{j=1}^{2}\left\{\frac{\partial Q_j^P}{\partial q_{A_j}} q_{A_j} + \left(\sum_{k=1}^{1}\frac{\partial Q_j^T}{\partial q_{A_k}} q_{A_k}\right)\right\}}{Q_{Miss}} = \frac{1\times 0.25 + 0.5\times 0.5 + 0\times 0.25}{0.875} = 0.5714 \quad (1st)$$

$$IC_B^M = \frac{\sum_{j=1}^{2}\left\{\frac{\partial Q_j^P}{\partial q_{B_j}} q_{B_j} + \left(\sum_{k=1}^{1}\frac{\partial Q_j^T}{\partial q_{B_k}} q_{B_k}\right)\right\}}{Q_{Miss}} = \frac{0\times 0.3 + 0.25\times 0.2 + 0.25\times 0.2}{0.875} = 0.1143 \quad (2nd)$$

Table 7.24 - Event combinations that contribute to the Mission Criticality importance measure value for each component in example mission *A*.

Event combination	Component A (failure phase)	Component B (failure phase)
$\overline{A}(0,2)\overline{B}(0,2)$	No	No
$A(0,1)\overline{B}(0,2)$	Yes	No
$A(1,2)\overline{B}(0,2)$	Yes	No
$\overline{A}(0,2)B(0,1)$	No	Yes
$\overline{A}(0,2)B(1,2)$	No	Yes
$A(0,1)B(0,1)$	Yes	No
$A(0,1)B(1,2)$	Yes	No
$A(1,2)B(0,1)$	No	Yes
$A(1,2)B(1,2)$	No	No

This shows that component *A* has the highest importance to the mission, whilst component *B* has a much lower importance. Table 7.24 and Table 7.25 show the event combinations that contribute to the Mission Criticality importance measure value of each component in example mission *A* and *B* respectively. They show that the event combinations that contribute to a component's Mission Criticality importance measure value are those that contribute to its Transitional or In-Phase Criticality importance in any phase of the mission.

Table 7.25 - Event combinations that contribute to the Mission Criticality importance measure value for each component in example mission *B*.

Event combination	Component A	Component B
$\overline{A}(0,2)\overline{B}(0,2)$	No	No
$A(0,1)\overline{B}(0,2)$	Yes	No
$A(1,2)\overline{B}(0,2)$	No	No
$\overline{A}(0,2)B(0,1)$	No	No
$\overline{A}(0,2)B(1,2)$	No	No
$A(0,1)B(0,1)$	Yes	No
$A(0,1)B(1,2)$	Yes	No
$A(1,2)B(0,1)$	Yes	No
$A(1,2)B(1,2)$	Yes	Yes

The relationship between IC_i^M and the sum of its contributing event combination probabilities, $P_{CEC_{IC_i^M}}$, is shown in Equation 7.60.

$$IC_i^M = \frac{P_{CEC_{IC_i^M}}}{Q_{Miss}} \qquad \textbf{7.60}$$

7.4 Summary and Conclusions

The literature review found that a vast number of importance measures have been developed for non-phased mission systems. Each definition was found to have been developed to show the contribution of a component to some particular aspect of system reliability. Each of the importance measures was analysed in detail, including application to components from a simple example system, to show exactly what it measures and how it defines a component as important. It was shown that some, such as the Criticality importance measure, give a good indication of the overall importance of a component to the system reliability, whilst others, such as the RAW importance measure, are useful tools for very specific types of analysis. In addition to those developed for measuring the contribution of individual components, importance measures developed to show the importance

of groups of components and the influence of component reliability on the importance of others were also found and discussed. For phased missions, only two importance measures were found to have been developed. These were interpretations of the Birnbaum and Criticality importance measures that had been previously developed, and are widely used, for the analysis of non-phased mission systems. The component and system failure time periods used in the phased mission importance measure definitions were discussed and explained. Each importance measure was then analysed and demonstrated through an application to components from an example phased mission system.

8 Phased Mission Importance Measures

8.1 Introduction

The literature review showed that six individual and five group component importance measures have been developed and are commonly used in the analysis of non-phased missions whereas only two individual component importance measures are available for phased missions. In addition, as will be shown later, the values assigned to components by the phased mission importance measures that have been developed lack the property of giving the precise influence the component has on system reliability. This situation is summarised in Table 8.1.

Table 8.1- Comparison between the development of importance measures for non-phased mission and phased mission systems.

	Non-phased missions	**Phased Missions**
Number of individual component importance measures found in the literature review	6	2
Number of group component importance measures found in the literature review	5	0
Do importance values show the precise influence a component has on system reliability?	Yes	No

A range of importance measures, similar to those available for non-phased mission systems, would be helpful to provide different perspectives on component importance since a particular importance measure may be the best choice for a particular analysis objective. The RRW importance measure, for example, can be used to find the component that provides the greatest scope for improving the reliability of the system. It is also a common practice when dealing with non-phased mission systems to average the importance values from multiple importance measures, for example from the Criticality and Fussell-Vesely importance measures, in order to gauge the overall relative importance of components. The current imbalance between the availability of importance measures for non-phased and phased mission systems therefore makes the optimisation of the reliability of a system more difficult if it operates in a phased mission. The development of phased mission definitions for each importance measure covered in the literature review, retaining the same interpretations, rectifies this situation.

The new importance measures introduced in this chapter are:

- Phased Birnbaum II
- Phased Criticality II
- Phased DIM
- Phased Risk Achievement Worth (Phased RAW)
- Phased Risk Reduction Worth (Phased RRW)
- Phased Fussell-Vesely
- Phased Criticality III

The first six of these are extensions to existing non-phased importance measures that were covered in the literature review. Of those, only the Birnbaum and Criticality importance measures have had phased mission definitions previously developed [2]. The new phased mission definitions of those provide alternatives with the advantage of giving values that show the precise influence a component has on the system reliability and can be used in predicting the outcome of component reliability alterations. Definitions to measure the importance of groups of, as well as individual, components are given and these are helpful in the common case of investigating optimisations to phased mission reliability that involve more than one component.

During the development of these importance measures it became clear that it would be useful if they could measure importance not only from component reliability in specific phases or a complete mission to system reliability in specific phases or a complete mission, but from and to any periods of the mission. Each importance measure has therefore been developed to allow this to the extent possible, since not all importance measures can be fully extended in this way due to limitations inherent to their definitions, and with the Phased Criticality III allowing full flexibility.

An analysis of the computational complexity of the Criticality importance measure interpretations for phased missions systems is included, showing that those presented in this chapter have lower computational expense than the definition covered in the literature review. The new definitions are therefore advantageous in applications such as real time analysis, the analysis of components in phased missions of many phases and for extension to the measurement of the importance of groups of components.

Finally, since it is often not the unreliability of the phased mission that is to be minimised but instead the reliability costs, a method that uses one of the newly developed importance measures to determine the reliability improvement to a component that minimises these costs is also presented.

The remainder of this chapter is organised as follows: Section 8.2 discusses the time periods used in the importance measure definitions, section 8.3 introduces the newly developed single component importance measures, sections 8.4 and 8.5 introduce the first and higher order component group importance measures respectively, section 8.6 covers the use of the importance measures in conditional analysis and the chapter finishes with section 8.7 that presents a method that can be used to minimise the reliability costs of a phased mission.

8.2 Component and system time periods in phased mission importance measures

The time periods for phased mission importance measures were described in [section 7.2.2]. As listed in that section, the possible component and system failure period combinations are importance of the failure of a particular component in a particular phase to:

1. System failure in a particular phase.
2. System failure on transition to a particular phase.
3. In-phase system failure in a particular phase.
4. System failure during the complete phased mission.

and the overall contribution of the failure of a particular component to:

5. System failure in a particular phase.
6. System failure on transition to a particular phase.
7. In-phase system failure in a particular phase.
8. System failure during the complete phased mission.

Combinations 2, 6, 7 and 8 can be measured by the existing phased mission importance measures introduced by Andrews [2]. However, there are no importance measure definitions for the measurement of the remaining 4 combinations and these also have practical uses. For example, an importance measure to measure combination 4 would be useful in the following scenario:

An engineer wishes to determine the component to which adding a heat shield will produce the greatest improvement in mission failures in a system that is exposed to heat stresses solely in phase 2. In such a case, the importance of the components failing in phase 2 to the mission reliability is of interest.

One of the developments to phased mission importance measures introduced in this chapter is therefore to expand their flexibility with regard to the phase periods that can be measured. The phased mission extensions to the importance measures that are introduced in this chapter have the following benefits with regard to time periods:

- They are able to measure importance from all 8 of the component phase importance periods listed above where such a definition has meaning. For some importance measures the measurement of certain time period combinations has no meaning due to the way in which the measure defines importance.
- A single form for the definition for measuring component importance to phase, in-phase and transitional system failure unlike the existing phased mission importance measures.

There are also situations that require importance measure analysis where the periods do not consist of a series of phases that begin and end at a phase boundary. For example, the consequences of aircraft engine failure within the climb phase are reduced during the latter stages when sufficient altitude has been obtained to render a

successful emergency landing highly probable. Another example would be a spacecraft that had the objective of releasing an exploration robot midway through the cruise phase of the mission. Thus the phases from an objective viewpoint, such as robot release, do not coincide with the reliability phases, such as cruise. In this case the analysis might concentrate on finding component importance to the mission up to that mid-phase point. Importance measures that allow the engineer to determine the importance of component failure to mission failure in the precise phases and periods of interest without restriction are therefore advantageous. The phased mission importance measures extended from non-phased mission definitions, both those by Andrews and those introduced in this chapter, are only able to analyse 'cause' periods that start and end at the phase boundaries. One solution might be to define more phases in the mission to create phase boundaries at the start and end of the time period of interest. However, that approach has the following drawbacks:

a. Adding additional phases increases the complexity of the model, for example, the number of nodes in a BDD model would increase substantially.
b. If multiple time periods are to be analysed then either multiple models with the appropriate phase boundaries for each or a single model with multiple additional phases would need to be formed.

Each of those drawbacks adds significantly to the computational expense of the analysis. The Phased Criticality III Importance Measure introduced in this chapter has the ability to measure importance over precise time periods whilst avoiding the need to change the system reliability model. Its advantages over other importance measure definitions with regard to time periods, in addition to the lower computational expense required for calculation, can be summarised as its:

- Ability to measure the importance of component failure in any period to system failure in any period of the mission - including those that do not coincide with phase boundaries.
- Single definition – i.e. it doesn't have separate definitions for in-phase, transitional and overall mission system failure. It therefore simplifies the analysis as there is no need to choose the appropriate definition of the importance measure for the analysis of specific time periods.

This flexibility will, for example, benefit the analysis of phased missions where the consequence of system failure varies within phases of the mission.

8.3 Importance Measure Definitions

In this section each of the new importance measures are introduced.

8.3.1 Introduction

The importance measures given in this chapter use the symbol *x* to denote the 'effect' period of the importance measure, where *x* is to be replaced by the time period of interest such as those from Table 2.2. For example, *x* is replaced by Q_j^P for the phase *j* in-phase failure period.

8.3.2 Phased Birnbaum II Importance Measure

A new phased mission definition of the Birnbaum importance measure is introduced in this chapter. Non-phased mission and phased mission definitions of this importance measures were covered in Section 7.2.2.3 and Section 7.3.3 of the literature review respectively. It is named the Phased Birnbaum II importance measure to distinguish it from the Phased Birnbaum importance measure introduced by Andrews [2] that was discussed in the previous chapter.

8.3.2.1 Motivation

The motivation for this definition was to provide an importance measure with the following two properties:

1. Its value always shows the ratio of the change in the system failure probability in a particular period to the change in the component failure probability in a particular period.
2. Its value does not depend on the system failure equation being in a certain form (e.g. Henley and Inagaki form).

Having property 1 is advantageous as it means that the exact reduction or increase in the system failure probability can be predicted for a given improvement or change in the failure probability of a component. It also means that reducing the probability of failure for a component ranked higher than another is certain to result in a larger reduction in the probability of system failure, thus helping with decisions aimed at increasing system reliability. The non-phased mission definition also has this property since for a non-phased mission the 'cause' and 'effect' periods are simply the operating time of the system used for the analysis. Whilst the Phased Birnbaum importance measures from Andrews will often give values close or the same as this property, it is not guaranteed and is sometimes significantly different. Property 2, which is again shared by the non-phased mission definition, is desirable since it means that the importance values measures can be calculated from any representation, for example the BDD technique which does not retain the Henley and Inagaki form.

8.3.2.2 Definition

In this section the Phased Mission Birnbaum II importance measure is defined for a component failure period that consists of a specific phase.

The period *x* 'effect' period Phased Birnbaum II importance measure with a phase *k* 'cause' period for component *i*, $IB2_{i_k}^{x}$, is defined as the partial derivative of the probability of system failure in period *x*, Q^x, with respect to the failure probability of component *i* during phase *k*, q_{i_k}, as shown by Equation 8.1. Note that Q^x can be in any form and the derivative, $\frac{\partial Q^x}{\partial q_{i_k}}$, is calculated exactly. It therefore measures the influence of failure of the component *i* in phase *k* on system failure in period *x* of the mission.

$$IB2_{i_k}^{x} = \frac{\partial Q^x}{\partial q_{i_k}} \qquad \textbf{8.1}$$

8.3.2.3 Interpretation of importance values

The importance value given by the Phased Birnbaum II importance measure gives the ratio of the change in the failure probability of the component in the 'cause' period to the change in the system failure probability in the 'effect' period. Whereas the valid values for the Birnbaum importance measures introduced in the literature review range from 0 to 1, the values for the phased mission interpretation introduced in this chapter range from -1 to 1 with higher values signifying greater importance. This section explains why they have this range and the interpretation of a negative value.

In non-phased coherent systems the survival of a component can have one of two effects on the probability of system failure:

1. Decrease the probability of system failure
2. No effect on the probability of system failure

For the Birnbaum importance measure, a component with effect 1 will have a positive importance and a component with effect 2 will have an importance of zero. This gives the Birnbaum importance measure its useful property of giving higher importance to components whose reliability has the greatest ability to influence the reliability of the system, and therefore lead to greater improvements in system reliability for a certain improvement in their own reliability.

In phased missions, the survival of a component through a certain period can have one of three effects on the probability of system failure in a certain period:

1. Decrease the probability of system failure in the time period of interest
2. No effect on the probability of system failure in the time period of interest.
3. Increase the probability of system failure in the time period of interest.

For example, consider the phased mission shown in Figure 8.1 and assume that each component has a probability of 0.1 of failing in each phase, which gives a probability of system failure in phase 2 of 0.18, from $Q_2 = p_{A_1} q_{B_{1,2}} = 0.9 \times 0.2 = 0.18$.

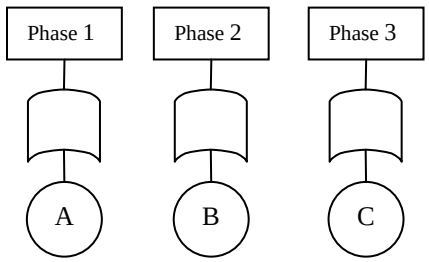


Figure 8.1 - A simple three phased mission.

Table 8.2 shows the effect that replacing each component with an alternative that is perfectly reliable in phase 1 has on the probability of the system failing in phase 2.

Table 8.2 – Effect of failure of a component in phase 1 on the probability of system failure in phase 2 for the example system described in Figure 8.1.

Component	Probability of system failure in phase 2.	Effect of increasing component reliability in phase 1 on probability of system failure in phase 2.
A	0.2 (an increase of 0.02).	Increases probability of system failure in phase 2.
B	0.09 (a decrease of 0.09).	Decreases probability of system failure in phase 2.
C	0.18 (no change).	Has no effect on probability of system failure in phase 2.

Reducing the failure probability of a component both reduces the probability of system failure due to failure modes including that component whilst simultaneously increasing the probability from those that do not include it. Intuitively this is due to the fact that reducing the probability of the system failure modes that include the component increases its exposure to the other system failure modes. Thus the reduction in the probability of system failure modes that include the component is always greater than or equal to the increase in the probability of the other failure modes since the increase in the latter is caused by the increased survival of the former. However, the net change in the probability of the system failure modes in the 'effect' period may be negative. When the net change in the probability of the system failure modes in the 'effect' period is negative we see effect 1, when it is 0 we see effect 2 and when it is positive we see effect 3.

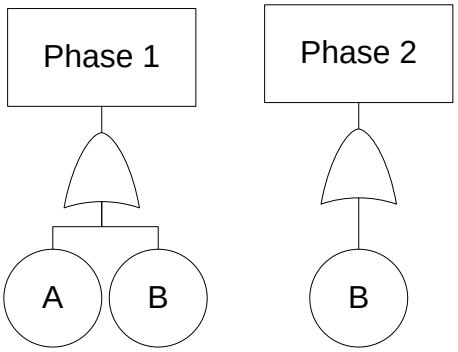
In coherent systems, whether or not they operate in a phased mission, reducing the probability always leads to an overall reduction in the probability of system failure when measured over the entire operating period of the system. For this reason, the values given by the non-phased mission Birnbaum importance measure to components in a coherent system are always non-negative. However it can lead to increases in the system failure probability in fractional time periods of the system operation, such as when measuring the impact of a component failure probability change on the system failure probability in a particular period. This is why reducing the failure probability of a component in a particular time period can lead to increased system failure probability in another time period such as in the example shown by Figure 8.1 and Table 8.2. The Phased Birnbaum II importance measure therefore gives a negative importance measure value when the net influence of decreasing the component's failure probability in the 'cause' period is an increase in the net probability of the system failure modes in the effect period.

8.3.2.4 Help in Decision Making

This section explains how the Phased Birnbaum II importance measure facilitates improved decision making compared to the Phased Birnbaum importance measure by accounting for the net influence a component's

'cause' period failure probability has on the probability of all system failure modes in the 'effect' period. An example of a decision often made by comparing Birnbaum importance measure values is deciding which component in a system should be replaced by a more reliable alternative to gain the maximum reduction in the system failure probability. The importance measure values given by the two phased mission definitions of the Birnbaum importance measure will be compared for two example phased missions to analyse their affect on the decision making process.

Table 8.3 shows, for example phased mission that is shown in Figure 8.2, the importance measure values for each component and 'effect' period with a 'cause' period of phase 1. The Phased Birnbaum importance measure gives the components equal importance values for the phase 1 in-phase 'effect' period whilst it does not define an importance value for the phase 2 in-phase 'effect' period. The decision maker might therefore assume that a reduction in the phase 1 failure probability of either component would have equal effect on the system reliability through the mission since the components are given equal importance by this importance measure for the 'effect' periods for which it is defined.



Component failure probabilities		
Component	**Phase 1**	**Phase 2**
A	0.1	0.1
B	0.1	0.8

Figure 8.2 – Example phased mission used to demonstrate the improved decision making abilities of the Phased Birnbaum II importance measure.

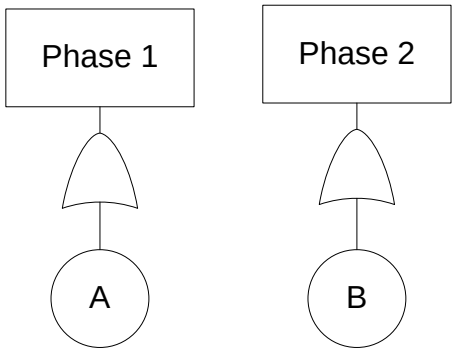
Table 8.3 - The importance measure values measuring the influence of phase 1 failure for each of the components in the phased mission shown in Figure 8.2.

	Component A		**Component B**	
	Phase 1 In-Phase	**Phase 2 In-Phase**	**Phase 1 In-Phase**	**Phase 2 In-Phase**
Phased Birnbaum	0.9	N/A	0.9	N/A
Phased Birnbaum II	0.9	-0.8	0.9	0

The Phased Birnbaum II importance measure also gives equal importance values to each component for the phase 1 in-phase 'effect' period, but for the phase 2 in-phase 'effect' period it gives a negative value to

component *A* and a value of zero to component *B*. This tells the decision maker that whilst both have an equal effect on the probability of system failure in phase 1, the improvement for component *A* is offset by a significant increase in the probability of system failure in phase 2. They would therefore be able to determine that reducing component *B*'s phase 1 failure probability would lead to the best overall improvement in mission reliability. This is confirmed by the fact that reducing the failure probability of component *A* in the 'cause' period, phase 1, by 0.05 results in a mission failure probability of 0.905 whereas the same reduction for component *B* results in a much lower mission failure probability of 0.865.

In the second example phased mission, shown in Figure 8.3, the second phase is instantaneous. This allows a full phase by phase comparison of the values given by the two Phased Birnbaum importance measure definitions as there are no phase 2 in-phase system failures and therefore the importance of a component with a phase 2 in-phase 'effect' period and phase 1 'cause' period, that is not defined by the Phased Birnbaum importance measure, can be ignored.



Component failure probabilities	
Component	**Phase 1**
A	0.1
B	0.7

Figure 8.3 – Another example phased mission used to demonstrate the improved decision making abilities of the Phased Birnbaum II importance measure.

Table 8.4 – The Phased Birnbaum and Phased Birnbaum II importance measure values for a phase 1 'cause' period for each of the components in the phased mission shown in Figure 8.3.

	Component A		**Component B**	
	Phase 1 In-Phase	**Phase 2 Transition**	**Phase 1 In-Phase**	**Phase 2 Transition**
Phased Birnbaum	1	0	0	0.9
Phased Birnbaum II	1	-0.7	0	0.9

The values given by the two phased mission interpretations of the Birnbaum importance measure are shown in Table 8.4. In this example, the Phased Birnbaum importance measure gives values of 1 and 0 to component *A* and values of 0 and 0.9 to component *B*. These values suggest that the influence of the two components is

similar, with a slightly greater overall importance given to component *A*. The values given by the Phased Birnbaum II importance measure however instead give a much higher overall importance to component *B*. The mission reliabilities after reductions of 0.05 in the phase 1 failure probabilities of components *A* and *B* are 0.715 and 0.685 respectively, confirming the higher relative influence of component *B*. The Phased Birnbaum importance measure has, for some situations, given importance values of 0 where the corresponding Phased Birnbaum II importance measure has given a negative value, such as the values for system failure on transition to phase 2 in the example shown in fig. 4. As discussed earlier, the non-phased Birnbaum importance measure gives an importance of 0 only in cases where the component's failure probability has no influence on the probability of system failure. This highlights another advantage of using the Phased Birnbaum II importance measure as it shows the decision maker the precise affects that changes to a component's reliability will have on the system reliability in each period of the system, whether that be an increase or decrease. For example, if component *A* in the system shown in Figure 8.3 were replaced with an alternative with better phase 1 reliability, the decision maker would know to expect and could plan for decreased system failures during phase 1 but increased failures in phase 2.

In summary the Phased Birnbaum II importance measure, in common with the widely used non-phased version, helps decision making by prioritising the components for which a given reduction in failure probability leads to the greatest reduction in the system failure probability and by showing the exact influence of the component in the 'cause' period to the system in the 'effect' period.

8.3.2.5 Practical Example

In this section a practical example of the use of the Phased Mission Birnbaum II is given.

The Scenario

A manufacturer is considering implementing a new burn-in process for the components in a safety system that will reduce infant mortalities (failures that occur early in the life of the component due to latent defects). The safety system contains three components and operates over three phases. The burn-in process is expensive and will only be applied to one of the components. An engineer wants to determine which component should be chosen based on the following:

1. It should not increase the probability of system failure in phase 2 as failures in that phase are hazardous and already at the maximum level specified by regulation.
2. Amongst those complying with rule 1 above, the component for which the burn-in process leads to the maximum reduction in phase 3 system failure probability should be chosen.

The phase failure probabilities for the three components are given in Table 8.5 which shows that the burn-in process reduces each component's phase 1 failure probability by 0.05. The phase fault trees for the system's mission are shown in Figure 8.4.

Table 8.5 – Safety system component phase failure probabilities.

Component	Phase 1 (with burn-in)	Phase 2	Phase 3
A	0.20 (0.15)	0.15	0.05
B	0.25 (0.20)	0.20	0.05
C	0.20 (0.15)	0.25	0.10

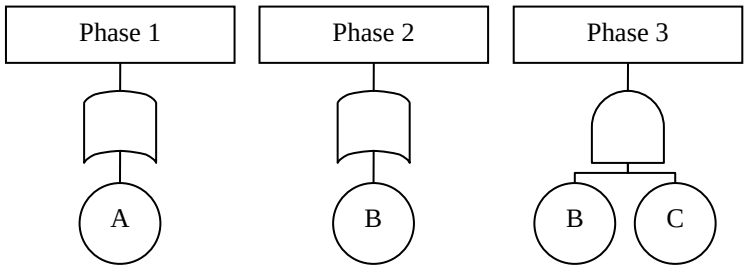


Figure 8.4 – Safety system phased mission fault trees.

The Solution

The engineer can easily find the solution to this problem using the Phased Birnbaum II importance measure. The first task is to determine those components for which applying the burn-in procedure will not lead to an increase in phase 2 system failure probability. Since the burn-in procedure reduces each component's phase 1 failure probability, it will lead to an increase in the probability of phase 2 system failure for any component with a negative Phased Birnbaum II importance for phase 2 'effect' and phase 1 'cause' periods. Only component *A* is assigned a negative importance by this and therefore only components *B* and *C* are suitable for the application of the burn-in process.

Table 8.6 – The phase 2 and phase 3 'effect' period Phased Birnbaum II importance measure values for phase 1 'cause' period for each component.

Component	Phase 2 'effect' period Phased Birnbaum II importance measure for phase 1 'cause' period	Phase 3 'effect' period Phased Birnbaum II importance measure for phase 1 'cause' period
A	-0.4500	-0.0275
B	0.8000	0.0000
C	0.0000	0.4400

The most suitable component from the remaining options, components *B* and *C*, is whichever leads to the greatest reduction in the probability of system failure in phase 3. This is the component with the highest phase 3 'effect' period Phased Birnbaum II importance measure for phase 1 'cause' period, since:

- That importance measure gives the ratio of the change in the phase 3 system failure probability to the change in the phase 1 component failure probability.
- The application of the burn-in process results in an identical change in the phase 1 failure probability for both components *B* and *C*.

The phase 3 'effect' period Phased Birnbaum II importance measure values for phase 1 'cause' period are 0 for component *B* and 0.44 for component *C*, as shown in column three of Table 8.6. The engineer should therefore choose to apply the burn-in procedure to component *C*.

The Birnbaum importance values for each component can be calculated when the design of a system is frozen, since they depend only on the system's reliability structure and are independent of any changes to the reliability of its components. These importance measure values could then be published with the system's design documentation and used in future reliability decisions such as in the example given here.

8.3.3 Phased Criticality II Importance Measure

This section introduces an extension to phased missions for the Criticality importance measure that was covered in Section 7.2.2.4 and Section 7.3.4 of the literature review. It is named the Phased Criticality II importance measure to distinguish it from the Phased Criticality importance measure that was introduced by Andrews [2] and because it is based on the Phased Birnbaum II importance measure that was introduced in the previous section.

8.3.3.1 Motivation

The main motivation for this new definition was to provide an importance measure that gives the precise influence of the failure probability of a component on the system failure probability, such that the system failure probability in the 'effect' period would be proportionally reduced by precisely the Criticality II importance measure value of the component when it is replaced by an alternative that is perfectly reliable in the 'cause' period (i.e. has no influence on the system failure). This is a useful property of the non-phased Criticality importance measure definition that the Phased Criticality importance measure lacks. A further motivation was to find a definition that requires the computation of fewer Birnbaum importance measures than the Phased Mission Criticality importance measure when the calculation of the overall Criticality of a component to a phased mission is required. This is a problem because the calculation of many Birnbaum importance measures has high computation expense.

8.3.3.2 Definition

The period x 'effect' period Phased Criticality II importance measure with a phase k 'cause' period for component i, $IC2_{i_k}^{x}$, is defined as the product of the corresponding Phased Birnbaum II importance measure, $IB2_{i_k}^{x}$, and the probability that component i fails in phase k, q_{i_k}, divided by the probability of system failure in

period *x*, as shown in Equation 8.2. The overall period *x* 'effect' period Phased Criticality II importance measure for component *i*, $IC2_i^x$, is defined by Equation 8.3. These importance measures give the influence of failure of component *i* in phase *k* on the system failure in period *x*. Therefore the reduction in the system failure probability in period *x* resulting from an elimination of component *i* failures in phase *k* (assuming its failure probability in other phases remains unchanged) and over the whole mission can be found by multiplying Q^x by the Phased Criticality II importance measure value. For example, a value of 0.25 indicates that the probability of system failure in period *x* would be reduced by 25% from its current value if component *i* failures in phase *k* or over the complete mission were eliminated for the definitions given in Equations 8.2 and 8.3 respectively. A value of -0.25, in contrast, indicates that the probability of system failure in period *x* would increase by 25% if the component *i* failures were eliminated.

$$IC2_{i_k}^x = \frac{IB2_{i_k}^x q_{i_k}}{Q^x} \quad \textbf{8.2}$$

$$IC2_i^x = \frac{\sum_{k=1}^{n} IB2_{i_k}^x q_{i_k}}{Q^x} \quad \textbf{8.3}$$

From Equation 8.3 it is apparent that the mission Phased Criticality II importance measure calculation for a component requires the calculation of *n* Birnbaum importance measures, whereas the number required by the equivalent Andrews Phased Criticality that is defined by Equation 7.59 is given by Equation 8.4. Since the Birnbaum importance measures involves the calculation of the change in the system failure probability for a change in a component reliability, it is clearly the most expensive operation in the Criticality importance calculations – the expense of the multiplication by the component failure probability and division by the system failure probability are insignificant in comparison. Since this is the most expensive operation in the Phased Criticality calculations, the new Phased Criticality definition represents a reduction in the complexity from quadratic, $O(n^2)$, to linear, $O(n)$, in the number of mission phases. For example, the calculation of the mission Phased Criticality II importance for a given component in a 10 phase mission would require the calculation of 10 Birnbaum importance measures compared to the 55 required by the Phased Criticality importance, a significant reduction.

$$N_{calcs} = \frac{n(n+1)}{2} \quad \textbf{8.4}$$

8.3.3.3 Interpretation of importance values

As with the Phased Birnbaum II importance measure, the range of this importance measure is -1 to 1, with higher values signifying greater importance. The value has the same interpretation as the non-phased definition, i.e. the influence of the component on the system failure probability. Therefore the system probability would reduce by the importance measure value (or increase for a negative importance value) if the component failures were eliminated.

8.3.3.4 Help in Decision Making

The Phased Criticality II importance measure has the advantage over the Phased Criticality importance measure that its value gives the precise influence of the component on the system failure probability. For example,

Table 8.8 shows the importance values assigned to the components by the two definitions for the components from the phased mission with the phase fault trees shown in Figure 8.4 and with component phase failure probabilities shown in Table 8.7.

Table 8.7 – Component phase failure probabilities for example phased mission.

Component	**Phase 1 (with burn-in)**	**Phase 2**	**Phase 3**
A	0.40	0.05	0.30
B	0.15	0.20	0.30
C	0.05	0.30	0.05

Since the importance value given to a component by the Phased Criticality II importance measure shows the change in the system failure probability in the 'cause' period due to the elimination of component failures in the 'effect' period, it is clear from the Phased Criticality II importance shown in

Table 8.8 that the probability of mission failure would be reduced by 18% if component *C* did not fail during the mission, a significant reduction.

Table 8.8 – Mission Phased Criticality importance measure values assigned to each component by the two Criticality importance measure definitions for phased missions.

Component	**Mission Phased Criticality Importance**	**Mission Phased Criticality II Importance**
A	0.600	0.33
B	0.400	0.40
C	0.013	0.18

Thus, it would be apparent to a decision maker analysing the Phased Criticality II importance values that component C, whilst not as influential as components *B* and *A* which are ranked as most and second most important respectively, has a large contribution to the current mission reliability. They would also see the exact influence each component has without the need for any additional calculations in the same way that they would with the non-phased Criticality definition. The Phased Criticality importance measure, however, assigns highest

importance to component *A* and assigns very little importance to component *C*. With the non-phased definition, removing the contribution to system failure of the highest ranked component results in the largest decrease in the probability of system failure, however this is not the case with the Phased Criticality, as if that were the case component *B* would be ranked highest as it is with the Phased Criticality II definition. The importance values assigned by the Phased Criticality definition also do not reflect the precise influence of the component on the system reliability as illustrated by the fact that it assigns an importance to component *C* that is around 2% of the value it assigns to component *A* when the relative reduction in mission unreliability that would result from removing the contribution of component *C* compared to that of component *A* is actually around 50%.

8.3.3.5 Practical Example

The Scenario

During the development of a new unmanned aerial vehicle (UAV), it is found that the initial prototype of its surveillance system often fails during the third phase of its test mission. Due to the nature of the intended operations for the UAV, it is critical that failures in that period are avoided. If the system fails earlier then the mission can be aborted without compromising the UAV and if it fails later then the most important surveillance data will have already been captured. The development team want to determine the influence that the phase 2 reliability of each component has on the phase 3 in-phase system failure probability.

The Solution

The team could calculate the phase 3 'effect' period in-phase Phased Criticality II importance measure with a phase 2 'cause' period for each component in the system. The importance value for each component would then correspond to the precise proportional reduction (or increase if negative) in the phase 3 in-phase system failure probability that would result from eliminating the possibility of it failing during phase 2. Ranking the components from highest importance to lowest would show the components with greatest influence and this information could be useful if they wanted to improve the in-phase 3 reliability of the system.

8.3.4 Phased DIM

The differential importance measure (DIM) that was developed by Borgonovo and Apostolakis [57] for measuring the importance of components from non-phased mission systems, gives the same rankings as the Birnbaum or Criticality importance measure dependant on the calculation method used. In this section, an equivalent definition is given for the measurement of the importance of components from phased mission systems.

8.3.4.1 Definition

The definition for the period x 'effect' period Phased DIM with a phase k 'cause' period for component i, $DIM_{i_k}^{x}$, is given by Equation 8.5, where Δq_{i_k} represents a variation in the phase k failure probability of component i and N_C is the total number of components in the system. The overall period x 'effect' period Phased DIM for component i, DIM_{i}^{x}, is defined by Equation 8.6.

$$DIM_{i_k}^x = \frac{\frac{\partial Q^x}{\partial q_{i_k}} \Delta q_{i_k}}{\sum_{s=1}^{N_C} \left| \frac{\partial Q^x}{\partial q_{s_k}} \Delta q_{s_k} \right|} \quad \textbf{8.5}$$

$$DIM_i^x = \frac{\sum_{k=1}^{n} \frac{\partial Q^x}{\partial q_{i_k}} \Delta q_{i_k}}{\sum_{s=1}^{N_C} \sum_{k=1}^{n} \left| \frac{\partial Q^x}{\partial q_{s_k}} \Delta q_{s_k} \right|} \quad \textbf{8.6}$$

Calculating the Phased DIM with Δq_{i_k} as a uniform fixed quantity for all components gives absolute importance values that are proportional to those from the Phased Birnbaum II importance measure. Alternatively, calculating the phased DIM with Δq_{i_k} as an identical proportion of the phase *k* failure probability of each component gives absolute importance values that are proportional to those from the Phased Criticality II importance measure. It therefore follows that the fixed and proportional quantity calculation methods result in identical component importance rankings as the Phased Birnbaum II and Phased Criticality II importance measures respectively. Since there are two ways to calculate the Phased DIM and each gives different absolute and relative importance values, it is important to state which method was used when presenting its results.

8.3.4.2 Interpretation of importance values

The value of the Phased DIM ranges from -1 to 1, with higher values signifying greater importance. As with the non-phased mission definition, the importance value assigned to a component by the Phased DIM gives its importance relative to that of all components from the same system. Since it gives relative importance and has no precise predictive qualities, the use of the Phased Birnbaum II or Phased Criticality II importance measures are advantageous when analysing the importance of single components. However, the Phased DIM does provide a basis for a useful group importance measure definition and this motivates its inclusion here.

8.3.5 Phased Risk Achievement Worth (RAW)

The Risk Achievement Worth (RAW) [54], discussed in Section 7.2.2.1, gives a measure of the achievement of the component in attaining the current system failure probability. A component given high importance by this measure is a key contributor in limiting the system failure probability to its current level and, for example, shows that replacing this component with a less reliable alternative would lead to a significant increase in the probability of system failure. This importance measure is extended to phased missions in this section with definitions for both the importance of the component in a particular phase and the importance of the component over the complete phased mission.

8.3.5.1 Definitions

The overall period *x* 'effect' period Phased RAW for component *i*, a_i^x, is defined by Equation 8.7.

$$a_i^x = sup\left\{ \frac{Q^x(q_{i_k} = 1)}{Q^x} : k \in \{1 \dots N\} \right\} \quad \textbf{8.7}$$

8.3.5.2 Interpretation of importance values

This importance measure has the same range as its non-phased counterpart, i.e. 0 to 1, with higher values indicating higher achievement. The importance value given to a component by this importance measure signifies the current achievement of the component as the ratio of the value of Q^x to Q^x if the component were to always fail at the point during the mission that maximised Q^x. It is therefore an indicator of the achievement, with respect to the 'effect' period system reliability, of the actual reliability compared to worse case reliability of the component. Thus a component with high importance through this measure is suppressing system failures during the 'effect' period compared to a component with worst case reliability.

8.3.5.3 Practical Example

A practical example of a situation in which this new importance measure would prove useful is given below.

The Scenario

A manufacturer needs to reduce the production cost of an electronic pump that operates in cycles of three phases (start-up, forward flow, reverse flow) so that it remains competitive in the market. The design team have produced new lower cost designs for two different components in the system, however since these parts are new and untested their reliability is unknown. Due to the nature of the application of the system it is critical that it does not fail and the manufacturer has decided that just one of the newly designed components will be introduced initially, thus minimising risk compared to introducing both simultaneously. The manufacturer wishes to determine which of the newly designed components should be chosen for introduction based on the requirement of avoiding system failures during the critical phase.

The Solution

By measuring the overall reverse flow phase 'effect' period RAW for the two components that are currently in the system and candidates for replacement by the new design, the manufacturer can determine which is achieving the least with respect to system failure probability in that phase (i.e. has the lowest RAW value) and should be replaced. In this way the manufacturer can avoid risking replacing a component that is performing well in keeping the probability of system failure at a low value during the reverse flow phase (i.e. that has a high RAW value). If, however, the manufacturer were able to determine the reliability of the newly designed components prior to replacement then they would be able to use the Phased Birnbaum II importance measure instead and determine the exact changes in the system failure probability during the reverse flow phase when either of the components were replaced, thereby eliminating the risk altogether.

8.3.6 Phased Risk Reduction Worth (RRW)

The Risk Reduction Worth (RRW) [54], discussed in Section 7.2.2.2, defines the importance of a component as its potential to reduce the probability of system failure. It therefore gives components with both high structural influence and high probability of failure the highest importance, since reducing their failure probability leads to large reductions in the system failure probability and they can also, potentially, be improved the most. Its extension to phased missions that is given in this section retains the same interpretation.

8.3.7 Definitions

The period *x* 'effect' period Phased RRW for component *i*, r_i^x, is defined by Equation 8.8, where $Q^x(q_i = 0)$ is defined as the probability of system failure in period *x* conditional on component *i* being perfectly reliable through the mission.

$$r_i^x = \frac{Q^x}{Q^x(q_i = 0)} \qquad \textbf{8.8}$$

8.3.7.1 Interpretation of importance values

As with the non-phased RRW, the range of the Phased RRW is 1 to ∞, with higher values signifying greater importance. The importance value assigned to a component gives the factor by which the probability of system failure in the 'effect' period would be reduced if the component were replaced by an alternative that were perfectly reliable. For example, a value of 3 indicates that the probability of system failure in the 'effect' period would fall to 33% of its current value if failures of the component were eliminated.

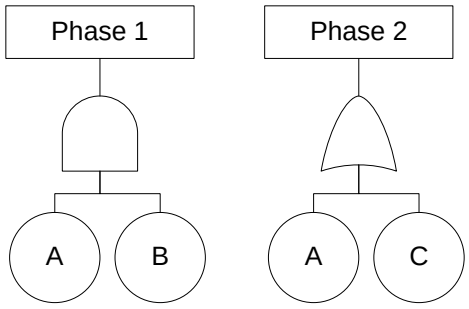
8.3.7.2 Practical Example

The Scenario

A chemical processing plant operator wishes to improve the reliability of a system that performs a two phase distillation process and currently has a higher than desired probability of failure in the final phase. The operator wants to improve the reliability of the system in that phase by improving the reliability of a component in the system. However, due to the costs involved with improving component reliability, the operator only wishes to proceed if the probability of system failure in the final phase can potentially be reduced by a factor of 1.5 through an improvement in the reliability of a single component.

The Solution

The phase 2 'effect' period Phased RRW importance values can be calculated for each component in the system and the operator can then proceed with the component reliability improvement plan if any has a value of greater than 1.5. For example, if the phased mission for the distillation process is represented by the phase fault trees and component failure probabilities shown in Figure 8.5, then the resulting phase 2 'effect' period Phased RRW importance values for components *A*, *B* and *C* are given by Equations 8.9 to 8.11. In this case, component *C* with an importance of 1.92 is the only component with an importance greater than 1.5 and therefore the only suitable candidate for reliability improvement due to the requirements of the operator. The value of 1.92 indicates that if component *C* were made perfectly reliability, the probability that the system fails during the second phase would be reduced by a factor of 1.92 from its current value.



Component failure probabilities		
Component	**Phase 1**	**Phase 2**
A	0.2	0.2
B	0.3	0.3
C	0.1	0.3

Figure 8.5 – Example phased mission for a distillation process.

$$r_A^{Q_2} = \frac{Q_2}{Q_2(q_A = 0)} = \frac{0.5}{0.4} = 1.25 \qquad \textbf{8.9}$$

$$r_B^{Q_2} = \frac{Q_2}{Q_2(q_B = 0)} = \frac{0.5}{0.44} = 1.14 \qquad \textbf{8.10}$$

$$r_C^{Q_2} = \frac{Q_2}{Q_2(q_C = 0)} = \frac{0.5}{0.26} = 1.92 \qquad \textbf{8.11}$$

8.3.8 Phased Fussell-Vesely Importance Measure

The non-phased definition Fussell-Vesely, given in Section 7.2.2.5, is a widely used importance measure and this provides an analogous definition for phased missions.

8.3.8.1 Definitions

The definition for the period *x* 'effect' Phased Fussell-Vesely importance measure for component *i*, FV_i^x, is given by Equation 8.12, where C_i^x is the set of minimal implicants for period *x* that include the failure of component *i* and Q^x is the probability of mission failure in period *x*.

$$FV_i^x = P\left(\bigcup_{c \in C_i^x} c\right) \div Q^x \qquad \textbf{8.12}$$

8.3.8.2 Interpretation of importance values

The Phased Fussell-Vesely importance measure has a range of 0 to 1, with higher values signifying greater importance, the same as the non-phased definition. The importance value assigned to a component indicates the probability that an implicant containing the failure of the component will occur during the 'effect' system failure period.

8.3.8.3 Practical Example

The Scenario

A pump control system that operates in a two phased mission with the phase fault trees shown in Figure 8.6 and component phase failure probabilities shown in Table 8.9, fails during the final phase. An engineer has been tasked with determining which components are most likely to have failed and caused an implicant to occur so that they can prioritise the testing and repairing of them.

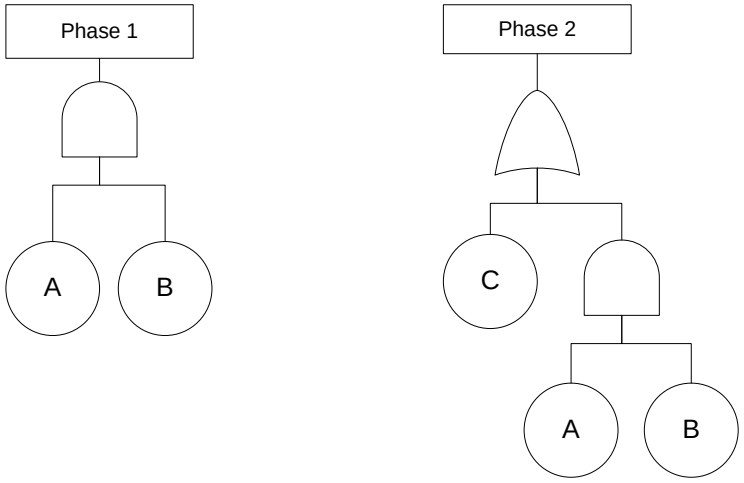


Figure 8.6 – Fault trees for pump control system mission.

Table 8.9 - Component failure probabilities for each phase of pump control system mission.

Component	Phase 1	Phase 2
A	0.05	0.1
B	0.2	0.1
C	0.1	0.1

The Solution

By calculating the phase 2 'effect' period Phased Fussell-Vesely importance measure for each component in the system and then ranking them from highest to lowest importance, the engineering team can test the highest importance components first, since these are the most likely to have failed and caused a phase 2 mission implicant to occur. From the phase fault trees and component failure probabilities for the pump control system phased mission, the phase 2 implicants are $A_2B_2 + \overline{A_1}\,\overline{B_1}C_{12}$ and the probability of system failure in that phase is 0.16. The phase 2 'effect' period Phased Fussell-Vesely importance value for each component in the system are given in

Table 8.10, showing that if the system fails in phase 2 then an implicant including the failure of component *C* is far more likely to have occurred than an implicant including the failure of component *B* or *C*.

Table 8.10 - Phase 2 Phased Fussell-Vesely importance for each component in the pump control system.

Component	Phase 2 Phased Fussell-Vesely
A	0.0625
B	0.0625
C	0.9500

8.3.9 Phased Criticality III Importance Measure

The Phased Criticality III Importance Measure is an entirely new importance measure definition that gives the same component rankings as the Phased Criticality II importance measure but has some useful advantages.

8.3.9.1 Motivation

The motivation for the development of this importance measure was the requirement for an importance measure that is able to measure component importance from a 'cause' period that does not coincide with a phase boundary. For example, consider a space craft that during its descent to earth experiences a period of extreme heat exposure for a short period on re-entry to the atmosphere of the earth whilst the reliability structure of many of its sub systems remains the same. In this case, measuring the importance for a 'cause' period consisting of only this re-entry period to a 'effect' period of the mission might be of interest in identifying components for which adding a heat shield would lead to the greatest improvement in mission reliability.

It also has a computational advantage when measuring component importance from a 'cause' period spanning multiple phases, such as when measuring overall importance to a phased mission, compared to the Phased Criticality II importance measure. This is beneficial when fast calculations are required or when analysing systems that operate in phased missions with high phase counts.

8.3.9.2 Definition

The previous Criticality importance measure definitions utilise the Birnbaum importance measure in their definitions, which is itself normally calculated by taking extreme values for the component failure probability (0 and 1). The Phased Criticality III instead uses a different and unique approach involving the modification of the component failure distribution so that its probability of failure over the 'cause' period is reduced by 1%. This not only allows any 'cause' period to be measured but also avoids the need to separately calculate and sum contributions from individual phase periods when a 'cause' period spans multiple phases as is necessary with the Phased Criticality and Phased Criticality II definitions. The probability that component *i* is failed at time *t* if its probability of failure between times t_1 and t_2 is reduced by 1% compared to its baseline distribution, denoted $q_i^-(t, t_1, t_2)$, is given by Equation 8.13 where $\psi_i(t)$ is its failure probability density function. The first integral term in Equation 8.13 represents the probability of failure before the 1% reduction period is reached, the second

term represents the probability of failure during that period and the third term represents the probability of failure after the period.

$$q_i^-(t, t_1, t_2) = \int_0^{\inf\{t,t_1\}} \psi_i(t)\, dt + \int_{t_1}^{inf\{t_2, sup\{t_1,t\}\}} 0.99\psi_i(t)\, dt + \int_{t_2}^{sup\{t,t_2\}} \psi_i(t)\, dt \qquad \textbf{8.13}$$

where $t_1 < t_2$.

The period *x* 'effect' period Phased Criticality III importance measure with 'cause' period t_1 to t_2, for component *i*, $IC3^x_{i_{t_1,t_2}}$, can then be defined as the change in the system failure probability during period *x* caused by a 1% reduction in the probability of component *i* failing during the 'cause' period, weighted by the probability of system failure during period *x*, multiplied by 100, as shown by Equation 8.14.

$$IC3^x_{i_{t_1,t_2}} = \left(\frac{Q^x - Q^x\left(q_i = q_i^-(t, t_1, t_2)\right)}{Q_x} \right) \times 100 \qquad \textbf{8.14}$$

Equation 8.14 shows that the Phased Criticality II importance measure requires only one calculation of a system failure change for a change in the failure probability of the component. This means that the algorithmic complexity of the overall importance of a component to the mission calculation is constant with the number of mission phases for the Phased Criticality III importance measure, $O(1)$, whereas it is quadratic, $O(n^2)$ and linear, $O(n)$, for the Phased Criticality and Phased Criticality II importance measures. The computational performance of the Criticality III importance measure is therefore far better than the other Phased Criticality importance measure definitions for systems that operate in phase missions consisting of high numbers of phases.

8.3.9.3 Interpretation of importance values

The range of this importance measure is -1 to 1, with higher values signifying greater importance. The values given by this importance measure have exactly the same meaning as those given by the Phased Criticality II importance measure, i.e. the influence of the component on the system failure probability, but the Phased Criticality III has the advantage that the measured 'cause' period can be any period in the mission and does not have to start and end at a phase boundary.

8.3.9.4 Practical Example

The Scenario

A certain rocket launching system has three components (A, B and C) and performs a three phased mission during the launch of a rocket. Each phase in the mission lasts 30 seconds and the reliability structure of the system in each phase is given by the phase fault trees shown in Figure 8.7.

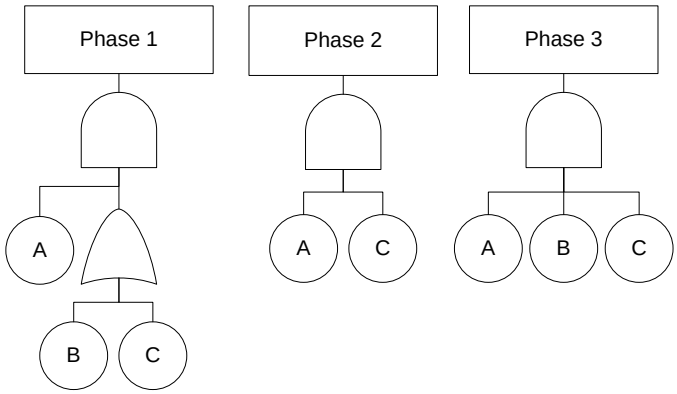


Figure 8.7 – Phase fault trees for the rocket launcher system.

The three components have the identical Weibull failure distribution given by Equation 8.15, where *t* is the component mission age in seconds. The hazard function for the components is plotted in Figure 8.8 and shows that the components suffer from a high hazard rate over the first 20 seconds of the mission due to infant mortality (e.g. failures due to defects in the manufacture of the components), after which the hazard rate is approximately constant.

$$q_A(t) = q_B(t) = q_C(t) = 1 - e^{-\left(\frac{t}{100}\right)^{0.8}} \qquad \textbf{8.15}$$

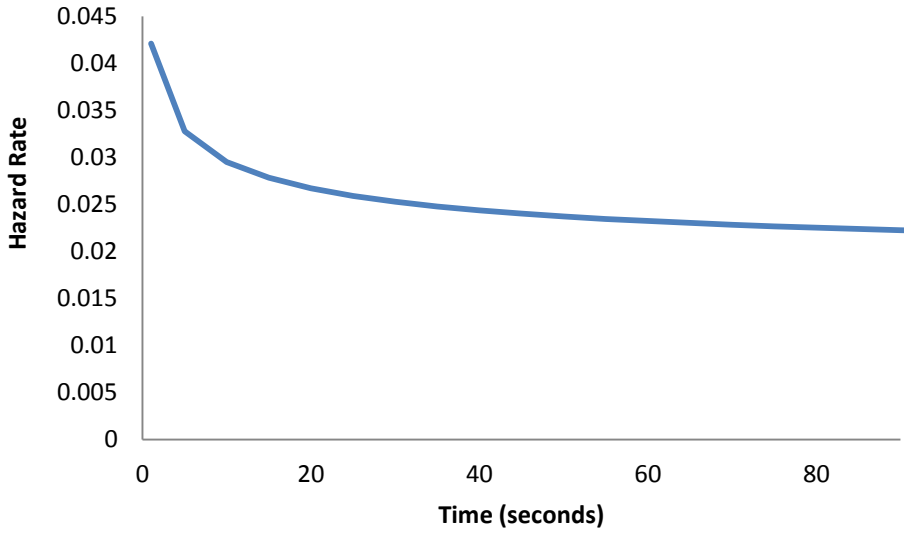


Figure 8.8 – Plot of hazard rate against time for components in the rocket launching system.

The designers of the system want to find out:

1. The influence the infant mortality of each component has on the mission reliability so that they can predict the reliability benefits that would result from the implementation of a burn-in program.
2. The overall influence of each component on the mission reliability.

The Solution

Since the ends of the infant mortality periods for the components do not coincide with a phase boundary, the Phased Criticality and Phased Criticality II cannot be used for the first analysis. The Phased Criticality III, however, is able to determine the precise contributions on the mission reliability from failures due to infant mortality alone. The mission 'effect' period Phased Criticality III importance measures for a 'cause' period from 0 to 20 seconds for components A, B and C are given by Equations 8.16 to 8.18. The importance values show that the infant mortality failures of component *A* has the greatest influence on the mission reliability since eliminating infant mortalities for that component would result in a 46.8% reduction in mission failures, compared to 17.3% and 31.2% reductions for component *B* and *C* respectively . The large importance values, particularly for component *A* and *B*, also show that infant mortalities are a significant contributor to the overall mission failure probability.

$$IC3_{A_{0,20}}^{Q_{Miss}} = \left(\frac{Q_{Miss} - Q_{Miss}\left(q_A = q_{\bar{A}}(t, 0{,}20)\right)}{Q_{Miss}}\right) \times 100 \qquad \textbf{8.16}$$

$$= \left(\frac{0.35140 - 0.34976}{0.35140}\right) \times 100 = 0.468$$

$$IC3_{B_{0,20}}^{Q_{Miss}} = \left(\frac{Q_{Miss} - Q_{Miss}\left(q_B = q_{\bar{B}}(t, 0{,}20)\right)}{Q_{Miss}}\right) \times 100 \qquad \textbf{8.17}$$

$$= \left(\frac{0.35140 - 0.35080}{0.35140}\right) \times 100 = 0.173$$

$$IC3_{C_{0,20}}^{Q_{Miss}} = \left(\frac{Q_{Miss} - Q_{Miss}\left(q_C = q_{\bar{C}}(t, 0{,}20)\right)}{Q_{Miss}}\right) \times 100 \qquad \textbf{8.18}$$

$$= \left(\frac{0.35140 - 0.35031}{0.35140}\right) \times 100 = 0.312$$

For the second part of the analysis, finding the overall influence of each component on the mission unreliability, the Phased Criticality III importance measure can also be used. The importance value calculations are given by Equations 8.19 to 8.21. This shows that component *A* has the highest and component *C* the lowest overall influence on the mission unreliability, with the failure of component *A* influential in all mission failures.

$$IC3_{A_{0,90}}^{Q_{Miss}} = \left(\frac{Q_{Miss} - Q_{Miss}\left(q_A = q_A^-(t,0,90)\right)}{Q_{Miss}}\right) \times 100 \quad \textbf{8.19}$$

$$= \left(\frac{0.35140 - 0.34789}{0.35140}\right) \times 100 = 1.00$$

$$IC3_{B_{0,90}}^{Q_{Miss}} = \left(\frac{Q_{Miss} - Q_{Miss}\left(q_B = q_B^-(t,0,90)\right)}{Q_{Miss}}\right) \times 100 \quad \textbf{8.20}$$

$$= \left(\frac{0.35140 - 0.35025}{0.35140}\right) \times 100 = 0.329$$

$$IC3_{C_{0,90}}^{Q_{Miss}} = \left(\frac{Q_{Miss} - Q_{Miss}\left(q_C = q_C^-(t,0,90)\right)}{Q_{Miss}}\right) \times 100 \quad \textbf{8.21}$$

$$= \left(\frac{0.35140 - 0.34890}{0.35140}\right) \times 100 = 0.714$$

Note that the equivalent calculations for the second analysis using the Phased Criticality importance measure instead would require the system reliability equations for in-phase and transitional failure for each phase and the calculation of 6 separate system to component reliability differentials – a significant increase in complexity. The reduction in computational expense realised by using the Phased Criticality III compared to the alternative criticality definitions is even more substantial when analysing systems operating in phased missions with higher phase counts.

8.4 First Order Group Importance Measures

Group importance measures give the importance of a group of components and can be used, for example, to compare the importance of different subsystems. They are very useful since the importance of a group of components does not necessarily correspond with the importance of the individual components in that group, i.e. a set of components may have higher importance as a group than another set of components that have higher importance individually. This section introduces a first order group importance measure that does not, therefore, account for the higher order interactions between components that lead to changes in the importance of components with the change in the reliability of others. It is therefore only suitable for ranking the importance of a group of components if those higher order interactions are likely to be small, such as when the purpose of the ranking is to determine which group of components is optimum for a small improvement in each of their reliabilities, with respect to improving the system reliability.

8.4.1 Phased Group DIM

The Group DIM that was discussed in Section 7.2.4.1 gives the importance of a group of components operating in a non-phased mission and is extended in this section to components from phased mission systems. The Phased Group DIM has the advantage that, like its non-phased mission counterpart, the calculation of the importance of a group of components is found directly from the sum of their individual DIM values.

8.4.1.1 Definition

The definition for the period *x* 'effect' period Phased Group DIM with a phase *m* 'cause' period for components *i*, *j* and *k*, $DIM^{x}_{\substack{i,j,k \\ m}}$, is given by Equation 8.22.

$$DIM^{x}_{\substack{i,j,k \\ m}} = DIM^{x}_{i_m} + DIM^{x}_{j_m} + DIM^{x}_{k_m} \qquad \textbf{8.22}$$

The overall period *x* 'effect' period Phased Group DIM for components *i*, *j* and *k*, $DIM^{x}_{i,j,k}$, is defined by Equation 8.23. As the definitions show, the Phased Group DIM value for a group of components is the sum of their equivalent individual Phased DIM values and, therefore, little additional computation is required.

$$DIM^{x}_{i,j,k} = DIM^{x}_{i} + DIM^{x}_{j} + DIM^{x}_{k} \qquad \textbf{8.23}$$

8.4.1.2 Interpretation of importance values

The range of this importance measure is -1 to 1, with higher values signifying greater importance. This importance value it assigns to a group of components signifies the relative importance of the components in that group compared to all components from the same system. Note that, since the Group Phased DIM is a first order group importance measure, the importance rankings are only valid for small variations in the 'cause' period failure probability of the components in each group. For example, when the individual component Phased DIM values are calculated for uniform variations to give individual component rankings that match those from the Phased Birnbaum II, reducing the 'effect' period failure probability of the components from a group ranked as most important by the Phased Group DIM may not lead to the maximal reduction in the 'effect' period system failure reliability, compared to the same action for lower ranked groups, if the component failure probability improvements is large.

8.4.1.3 Practical Example

The Scenario

A certain system involved in the refinement of petroleum operates in a phased mission. The Phased DIM values for the individual components in the system have already been calculated and the operator wants to quickly compare the relative importance of several of its subsystems to the final phase of the mission. The operator will use the information to determine which subsystem will be subjected to increased preventive maintenance, which is expected to result in a small decrease in the failure probability of its components.

The Solution

The operator can quickly find the overall final phase 'effect' period Group Phased DIM for each subsystem by summing the individual overall final phase 'effect' period Phased DIMs of its constituent components. The subsystems can therefore be ranked and their relative importance compared with minimal calculations necessary and, since the expected reliability improvement for the components in the selected subsystem is small, these rankings are likely to be accurate and unaffected by any higher order component interactions.

8.5 Higher Order Group Importance Measures

Higher order group importance measures account for the interactions between components within the group to give precise values for their importance and provide information that is very useful for optimising the reliability of a phased mission when the reliability of multiple components is to be varied. The disadvantage compared to the Group DIM is that they have a comparatively higher computational expense. The higher order group importance measures that have been developed for non-phased missions and were discussed in Section 7.2.5, namely the Group RAW, Group RRW, Joint Importance Measure and Differential Importance Measure II are all extended to phased missions in this section. In addition, an extension to the Phased Criticality III importance measure for component groups is introduced.

8.5.1 Phased Group RAW

The Group RAW for non-phased mission systems, discussed in Section 7.2.5.2, measures the achievement of a group of components in attaining the current system unreliability. An equivalent definition for the analysis of components from phased mission systems is introduced in this section.

8.5.1.1 Definitions

The period *x* 'effect' period Phased Group RAW for the group of components *i*, *j* and *k*, $a^x_{i,j,k}$, is defined by Equation 8.24, where $Q^x(q_{i_l}=1, q_{j_m}=1, q_{k_n}=1)$ is the system failure probability in period *x* when components *i*, *j* and *k* are failed in phased *l*, *m* and *n* respectively.

$$a^x_{i,j,k} = sup\left\{\frac{Q^x(q_{i_l}=1, q_{j_m}=1, q_{k_n}=1)}{Q^x} : l,m,n \in \{1 \dots n\}\right\} \qquad \textbf{8.24}$$

8.5.1.2 Interpretation of importance values

The values for the Phased Group RAW range from 0 to 1, with higher values indicating higher achievement. The value assigned to a component group by the period *x* Phased Group RAW is the ratio of Q^x if the components in the group were to fail in the set of phases that maximised it, to the current Q^x. It therefore gives the achievement of the group, with respect to Q^x, in comparison to their worst case reliability.

8.5.1.3 Practical Example

The Scenario

An emergency sprinkler system performs a three phased mission in response to a fire alert. The system is constructed by the manufacturer by integrating three subsystems, each of which contains a number of components and is procured from a separate supplier. The manufacturer wishes to evaluate the achievement of each subsystem in achieving the current mission reliability, so that any found to have a poor achievement can be investigated and an alternative equivalent subsystem, perhaps from a different supplier, considered.

The Solution

The manufacturer can apply the Phased Group RAW to the group of components from each subsystem and then analyse the achievement values allocated. If a low achievement value is assigned to the group of components from a particular subsystem then this indicates that it has close to the worst case reliability with respect to the three phased mission and should be investigated. There would be little risk of increasing the mission unreliability by replacing a subsystem with low achievement with an alternative subsystem.

8.5.2 Phased Group RRW

In this section, an extension of the non-phased mission Group RRW, discussed in Section 7.2.5.3, to a definition for the analysis of components from phased mission systems is introduced. This definition gives a measure of the maximum possible improvement in system failure probability that could be achieved through improving the reliability of a group of components.

8.5.2.1 Definition

The period x 'effect' period Phased RRW for the group of components i, j and k, $r^x_{i,j,k}$, is defined by Equation 8.25 where $Q^x(q_i = 0, q_j = 0, q_k = 0)$ is the probability of system failure in period x if all three components were perfectly reliable.

$$r^x_{i,j,k} = \frac{Q^x}{Q^x(q_i = 0, q_j = 0, q_k = 0)} \qquad \textbf{8.25}$$

8.5.2.2 Interpretation of importance values

The values assigned to a group of components by this importance measure range from 0 to ∞, with higher values indicating greater importance. The value assigned to a group is the ratio of the current Q^x value to the value of Q^x if the group of components were perfectly reliable. It therefore gives the maximum improvement in Q^x that can be attained through the reliability of the components in that group.

8.5.2.3 Practical Example

The Scenario

A company is looking for a supplier of a de-icing system that meets a specified reliability requirement over the five phased mission in which it will operate. A manufacturer is interested in supplying their system but its current design has a mission reliability that is 80% of the required value. The manufacturer has identified several subsystems within the system that contribute to mission failures and wishes to determine if improving the reliability of one of those subsystems could result in a system that meets the mission reliability requirement of the supplier.

The Solution

The manufacturer could apply the mission 'effect' period Phased Group RRW to each of the subsystem component groups and then rank those groups by importance. Each of those with a value of 1.25 or greater would indicate that sufficient improvement to the reliability of that subsystem would result in a system that meets the mission reliability requirement.

8.5.3 Phased Joint Birnbaum II Importance Measure

The Phased Joint Birnbaum II importance measure gives the impact of the reliability of one component on the Phased Birnbaum II importance of the other in a pair of components. This can help to determine which components to improve the reliability of when the goal is to improve the reliability of the system. A pair of components with negative joint importance indicates that improving the reliability of one component would result in a greater system reliability improvement from the reliability improvement of the other and they should, therefore, be improved simultaneously if possible. On the contrary, improving the reliability of one component from a pair with positive joint importance would result in reduced impact on the system reliability from the improvement of the other and therefore their simultaneous improvement will have a lesser influence on the system reliability than if the component reliability interactions were absent.

8.5.3.1 Definition

The period *x* 'effect' period Phased Joint Birnbaum II importance with a phase *m* 'cause' period for components *i* and *j*, $JIB2_{\underset{m}{i,j}}^{x}$, is given by Equation 8.26. It therefore shows the change in the period *x* 'effect' period Phased Birnbaum II importance with phase *m* 'cause' period for a component when the phase *m* reliability of another is changed.

$$JIB2_{\underset{m}{i,j}}^{x} = \frac{\partial^2 Q^x}{\partial q_{i_m} \partial q_{j_m}} \qquad \textbf{8.26}$$

8.5.3.2 Interpretation of importance values

The range of values for this importance measure is -1 to 1 with higher values indicating a stronger correlation between the Phased Birnbaum II and reliability of the components in the pair. The value indicates the rate of change in the importance of one component with the change in the reliability of the other, with a positive value indicating that the correlation is positive (i.e. reducing the reliability of one reduces the importance of the other and vice versa) and a negative value indicating a negative correlation.

8.5.3.3 Practical Example

The Scenario

A certain aircraft landing gear actuator system operates in a phased mission over a flight consisting of two 60 second phases, stowaway and extension, for which the fault trees are shown in Figure 8.9. The components in the system have the exponential failure distributions given by Equations 8.27 to 8.29, where *t* is the component mission age in seconds.

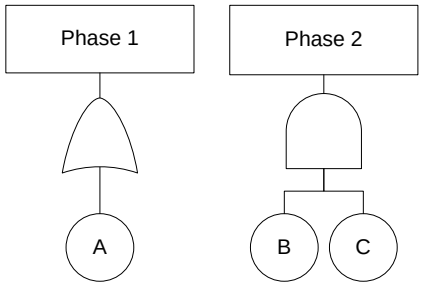


Figure 8.9 – Phase fault tree for aircraft landing gear actuator system example.

$$q_A(t) = 1 - e^{-0.01t} \quad \textbf{8.27}$$

$$q_B(t) = 1 - e^{-0.03t} \quad \textbf{8.28}$$

$$q_C(t) = 1 - e^{-0.02t} \quad \textbf{8.29}$$

The system currently has a high probability of failure in phase 2 and a reliability engineer wishes to determine how to reduce this by improving the phase 2 reliability of its components.

The Solution

In order to determine which components the reliability engineer should improve the phase 2 reliability of in order to increase the phase 2 reliability of the system, the first step is to calculate the phase 2 ‘effect’ period Phased Birnbaum II importance with a phase 2 ‘cause’ period for each component, as shown by Equations 8.30 to 8.32. These importance values show that a given reduction in the phase 2 failure probability of either

component *B* or *C* will result in an improvement in the system reliability of approximately half that amount, e.g. a reduction of 0.05 in the phase 2 failure probability for component *B* will result in a 0.025 reduction in the system failure probability in the same phase.

$$IB2_{A_2}^{x} = \frac{\partial Q_2}{\partial q_{A_2}} = 0 \qquad \textbf{8.30}$$

$$IB2_{B_2}^{x} = \frac{\partial Q_2}{\partial q_{B_2}} = 0.499 \qquad \textbf{8.31}$$

$$IB2_{C_2}^{x} = \frac{\partial Q_2}{\partial q_{C_2}} = 0.534 \qquad \textbf{8.32}$$

However, if the simultaneous improvement of multiple components is being considered then it is important to also determine how the importance of each component varies with the reliability of others. The reliability engineer could therefore also calculate the phase 2 'effect' period Phased Joint Birnbaum II importance with a phase 2 'cause' period for each component pair, as shown by Equations 8.33 to 8.35. The values of 0 for the component pairs that include component *A* show that its phase 2 reliability is independent of the phase 2 Phased Birnbaum II importance of the other components (and, similarly, that its importance is independent of their reliability). However, the high negative value for component pair *A* and *B* shows that improving the phase 2 reliability of either of those components increases the phase 2 Phased Birnbaum II importance of the other. This indicates that it would be beneficial to improve the phase 2 reliability of both components *B* and *C* in tandem, since each reduction in the phase 2 failure probability of one component not only reduces the phase 2 system failure probability but also increases the amount by which a reduction in the phase 2 failure probability of the other will reduce the phase 2 system failure probability.

$$JIB2_{\substack{A,B \\ 2}}^{Q_2} = \frac{\partial^2 Q^{Q_2}}{\partial q_{A_2} \partial q_{B_2}} = 0 \qquad \textbf{8.33}$$

$$JIB2_{\substack{A,C \\ 2}}^{Q_2} = \frac{\partial^2 Q^{Q_2}}{\partial q_{A_2} \partial q_{C_2}} = 0 \qquad \textbf{8.34}$$

$$JIB2_{\substack{B,C \\ 2}}^{Q_2} = \frac{\partial^2 Q^{Q_2}}{\partial q_{B_2} \partial q_{C_2}} = -0.549 \qquad \textbf{8.35}$$

8.5.4 Phased Group Criticality III Importance Measure

The Phased Criticality III can also be extended to the measurement of the importance of groups of components and the definition for this, the Phased Group Criticality III importance measure, is given in this section. This is a very useful importance measure as it gives the precise influence that the failures of the group of components in a specified 'cause' period have on the system reliability in a specified 'effect' period.

8.5.4.1 Definition

The period *x* 'effect' period Phased Group Criticality III importance with 'cause' period t_1 to t_2 for components *i*, *j* and *k*, $IC3^{x}_{\substack{i,j,k \\ t_1,t_2}}$, is defined by Equation 8.36, where $Q^x\left(q_i = q_i^-(t,t_1,t_2), q_j = q_j^-(t,t_1,t_2), q_k = q_k^-(t,t_1,t_2)\right)$ is the probability of system failure in period *x* when the probability of failure for each of the components is reduced by 1% in the 'cause' period.

$$IC3^{x}_{\substack{i,j,k \\ t_1,t_2}} = \left(\frac{Q^x - Q^x\left(q_i = q_i^-(t,t_1,t_2), q_j = q_j^-(t,t_1,t_2), q_k = q_k^-(t,t_1,t_2)\right)}{Q^x}\right) \times 100 \qquad \textbf{8.36}$$

Note that if an equivalent importance measure were to be defined using the Phased Criticality or Phased Criticality II importance measure then it would require the summation of the contributions from all possible different 'cause' failure phase combinations between the components in the group and would therefore entail significant computational expense if the component group were large or the 'cause' period spanned multiple phases (such as when measuring overall contribution of a component group to phased mission reliability). The Phased Criticality III importance measure avoids this and allows the importance of a group of components to be calculated with approximately the same computational expense as for a single component.

8.5.4.2 Interpretation of importance values

The range of this importance measure is -1 to 1 with higher values indicating higher importance. The importance value assigned to a group of components indicates the influence that the group of components has on Q^x, i.e. the proportional reduction (or increase if negative) in Q^x that would occur if the components in the group were perfectly reliable during the 'cause' period.

8.5.4.3 Practical Example

The Scenario

An anti-aircraft cruise missile system operates in a phased mission that starts with launch and ends with impact at its target. The high g-forces experienced by the subsystems of the missile, caused by sharp manoeuvring in the last part of the final phase of the mission prior to impact, induce high component failure rates. The design team of the missile wish to determine the influence of the reliability of each subsystem during this final stage of the final phase on the overall mission reliability.

The Solution

The design team could apply the mission 'effect' period Phased Group Criticality III with the high g-force 'cause' period for the group of components from each subsystem. The value assigned to each group would indicate the exact influence of the failures of the components in that group during the high g-force period have on the mission reliability. Ranking the subsystems by this importance measure would, for example, show which subsystems would benefit the mission reliability if their robustness to g-forces were improved.

8.5.5 Phased DIM II

The DIM^{II} importance measure, discussed in Section 7.2.5.4, defines the importance of a pair of components as the influence that a variation in their reliabilities has on the system reliability relative to that of all component pairs in the system. It accounts for both the first and second order component reliability interactions and can therefore be used, for example, to determine the pair of components in a system for which a reliability improvement would result in the maximal system reliability improvement. Definitions for the DIM^{II} for application to components in phased mission systems are introduced in this section.

8.5.5.1 Definition

The definition for the period *x* 'effect' period Phased DIM^{II} with a phase *m* 'cause' period for components *i* and *j*, $DIM^{II}{}^{x}_{m\,i,j}$, is given by Equation 8.37, where $\Delta Q^{x}_{m\,i,j}$ is defined by Equation 8.38 as the change in the probability of system failure during period *x* due to small variations, Δq_{i_m} and Δq_{j_m}, in the phase *m* failure probability of components *i* and *j* respectively.

$$DIM^{II}{}^{x}_{m\,i,j} = \frac{\Delta Q^{x}_{m\,i,j}}{\sum_{s=1}^{N_c}\sum_{t>s=1}^{N_c}\left|\Delta Q^{x}_{m\,s,t}\right|} \qquad \textbf{8.37}$$

$$\Delta Q^{x}_{m\,s,t} = \frac{\partial Q^{x}}{\partial q_{s_m}}\Delta q_{s_m} + \frac{\partial Q^{x}}{\partial q_{t_m}}\Delta q_{t_m} + \frac{\partial Q^{x}}{\partial q_{s_m}\partial q_{t_m}}\Delta q_{s_m}\Delta q_{t_m} \qquad \textbf{8.38}$$

Correspondingly, the definition for the overall period *x* 'effect' period Phased DIM^{II} importance measure for components *i* and *j*, $DIM^{II}{}^{x}_{i,j}$, is given by Equation 8.39.

$$DIM^{II}{}^{x}_{i,j} = \frac{\sum_{m=1}^{n}\Delta Q^{x}_{m\,i,j}}{\sum_{s=1}^{N_c}\sum_{t>s=1}^{N_c}\sum_{m=1}^{n}\left|\Delta Q^{x}_{m\,s,t}\right|} \qquad \textbf{8.39}$$

8.5.5.2 Interpretation of importance values

The range of this importance measure is -1 to 1 with higher values signifying greater importance. The importance value assigned to a pair of components signifies its relative importance in comparison to that of all component pairs from the same system.

8.5.5.3 Practical Example

The Scenario

A manufacturer has been informed by a supplier that a recent batch of components are suspected of being faulty and have a failure rate that is potentially much greater than normal. Two of these parts have been used in a certain system the manufacturer produces and that operates in a phased mission. They want to know if the mission reliability of that system is likely to be strongly affected, in which case they will issue a recall.

The Solution

The manufacturer could calculate the overall mission period Phased DIM II importance of the pair of components to determine if that pair of components, relative to all pair in the system, have a strong influence on the mission reliability. A high importance value will indicate that the mission reliability could be greatly increased and a recall may therefore be necessary.

8.6 Conditional Importance Measures

The importance of components and groups of components, as measured by the importance measures that have been introduced in this chapter, can be updated as a phased mission progresses to account for:

- The fact that the component failure combinations that signify mission failure up to the point reached cannot have occurred.
- The known working or failed status of components with revealed failures.

The updating of component importance as a mission progresses has many uses. For example, the monitoring of component importance through a phased mission could be used to determine which components should receive additional preventive maintenance at any point – a strategy that would maximise the mission reliability that can be achieved with limited maintenance resources.

The conditional importance of a component or group of components can be found using any of the importance measures introduced in this chapter. This is achieved by replacing all occurrences of system failure in the 'effect' period, Q^x, that appear in the importance definition with its equivalent that is conditional on the point in the mission reached and the status of the components, working or failed, for those with revealed failures. For example, the period x 'effect' period Phased Birnbaum II importance with a phase k 'cause' period for component i conditional on the system surviving to the end of phase 3, component s working up to that point and component t having failed in phase 2, $IB2_{i_k}^{x}\left(T > t_3, q_{s_3} = 0, q_{t_2} = 1\right)$, would be given by Equation 8.40, where T is the mission failure time and $Q^x\left(T \geq t_3, q_{s_3} = 0, q_{t_3} = 1\right)$ is the period x system failure probability based on those conditions.

$$IB2_{i_k}^{x}\left(T > t_3, q_{s_3} = 0, q_{t_2} = 1\right) = \frac{\partial Q^{x}\left(T \geq t_3, q_{s_3} = 0, q_{t_3} = 1\right)}{\partial q_{i_k}} \quad \textbf{8.40}$$

The Phased Criticality III importance measure is particularly well suited to use as a conditional importance measure where frequent updates of importance values are required due to its high computational efficiency.

8.7 Reliability Cost Minimisation

Often the optimisation of a phased mission system requires a minimisation of its reliability costs, influenced by both the probability and cost of failures. The importance measures discussed so far, both in this chapter and the literature review, are concerned solely with the reliability and do not account for costs. In this section, a method for using the Phased Criticality III importance measure to minimise reliability costs is presented. Using this method it is possible to find the exact amount by which the reliability of a component in a certain period should be improved to minimise the costs due to reliability over a complete phased mission.

8.7.1 Reliability Costs

The different costs associated with the reliability of a component and phased mission system are discussed in this section.

8.7.1.1 Component costs

Improving the reliability of a component may incur some one-off costs such as:

- Investigating how the component can be improved.
- Finding a supplier of a more reliable alternative to the current component.
- Re-designing and testing the component.

These costs are only incurred a single time and are independent of how many phased missions are carried out with the improved component in the system. The cost per mission is therefore the total one-off improvement costs divided by the number of times the system carries out the phased mission. In addition, these costs may not vary linearly with the magnitude of the reliability improvement. For example a 10% reduction in the failure probability of a component may involve one-off costs that are more than twice those necessary to attain a 5% reduction. The per mission one-off costs of achieving an *x*% reduction in the probability of component *i* failing in period t_1 to t_2 of the mission is denoted by C_{a_i}.

There may also be a change in the costs directly associated with the failure of the component, not including those incurred at the system level. This includes the cost of repairing or replacing the component after a phased mission to restore it to the working state for the next phased mission undertaken by the system. For example, the direct costs of failure for a component might increase as its reliability is improved if this improvement is achieved by manufacturing it from higher quality but more expensive materials. As with the one-off costs, the increase in the cost of component failure may not vary linearly with the magnitude of the reliability improvement. The expected direct cost of failure for component *i* in a phased mission, $\mathrm{E}\left(C_{d_i}\right)$, is given by Equation 8.41, where C_{d_i} is the direct cost of component *i* failing.

$$\mathrm{E}(C_{d_i}) = q_i \times C_{d_i} \qquad \textbf{8.41}$$

The change in the expected direct costs of component *i* failure due to an *x*% reduction in the probability of it failing in period t_1 to t_2 of the mission, $\Delta\mathrm{E}(C_{d_i})$,is given by Equation 8.42, where $q_{i_{t_1,t_2}}$ is the probability of component *i* failing between times t_1 and t_2 prior to the reliability improvement and $C_{d-prior_i}$ and $C_{d-after_i}$ are the direct costs of component *i* failing prior and after the reliability improvement respectively. A positive and negative value for $\Delta\mathrm{E}(C_{d_i})$ indicate an increase and reduction in the expected direct cost of component failure over a phased mission respectively.

$$\Delta\mathrm{E}(C_{d_i}) = q_{i_{t_1,t_2}}\left(C_{d-after_i}\left(1-\frac{x}{100}\right) - C_{d-prior_i}\right), \qquad \textbf{8.42}$$

where $0 < x < 100$.

8.7.1.2 System Costs

The system costs are the costs incurred from system failures during a phased mission and may be related to direct economic factors such as lost production as well as costs assigned to consequences such as loss of lives. Unique to phased missions is the possibility for a variation in the cost of system failure with the period in which it occurs. For example, the failure of an aircraft engine during the initial period of the climb phase would have much higher expected costs, such as catastrophic loss of the aircraft, than if the same failure were to occur during the final taxi phase where they may be limited to those related to the repair of the engine. The phased mission can therefore be split into a set of consecutive time periods, $p_1$to p_m, where *m* is the total number of periods, each of which is associated with the cost of mission failure in that period. These costs are denoted $c_{p_1}$to c_{p_m}, where $c_{p_1}$is the cost of failure in period p_1 and so on. Two special cases are worth noting:

1. If the cost of mission failure depends only on the phase in which it occurs then these periods will coincide with the phases.
2. If the cost of mission failure is independent of the phase in which failure occurs then there will be a single period that spans the complete mission.

The expected cost due to system failure in a phased mission, $\mathrm{E}(C_{sys})$, is therefore given by Equation 8.43, where Q^{p_n} is the probability of mission failure in period p_n.

$$\mathrm{E}(C_{sys}) = \sum_{n=1}^{m} c_{p_n} \times Q^{p_n} \qquad \textbf{8.43}$$

The change in the expected in the costs due to system failure in a phased mission, $\Delta\mathrm{E}(C_{sys})$, due to an *x*% reduction in the probability of component *i* failing in period t_1 to t_2 of the mission, is given by Equation 8.44

where $IC3_{i_{t_1,t_2}}^{p_n}$ is the p_n 'effect' period Phased Criticality III importance measure with 'cause' period t_1 to t_2 for component *i*. A positive value for $\Delta\mathrm{E}(C_{sys})$ indicates an increase in the expected cost of system failure over a phased mission, whilst a negative value indicates a decrease.

$$\Delta\mathrm{E}(C_{sys}) = -x \sum_{n=1}^{m} c_{p_n} \times \frac{IC3_{i_{t_1,t_2}}^{p_n}}{100}, \qquad \textbf{8.44}$$

where $0 < x < 100$.

8.7.2 Overall Reliability Costs

The change in the expected total reliability cost of a phased mission, $\Delta\mathrm{E}(C_{total})$, that results from a *x*% reduction in the probability of component *i* failing over the period t_1 to t_2 of the mission is given by Equation 8.45.

$$\Delta\mathrm{E}(C_{total}) = C_{a_i} + \Delta\mathrm{E}(C_{d_i}) + \Delta\mathrm{E}(C_{sys}) \qquad \textbf{8.45}$$

A negative value for $\Delta\mathrm{E}(C_{total})$ indicates a reduction in the expected overall reliability costs whilst a positive value indicates an increase.

8.7.3 Finding the optimum component reliability improvement

Equation 8.45 allows the change in the expected overall expected reliability cost to be found for any percentage improvement in the reliability of a component, assuming the two cost functions that give C_{a_i} and C_{d_i} for a given x% reduction in the probability of component *i* failing over the period t_1 to t_2 of the mission have been estimated. The optimum component reliability improvement percentage, and its corresponding reduction in reliability cost, can therefore be found by plotting $\Delta\mathrm{E}(C_{total})$ against the percentage reliability improvement.

The rocket launcher phased mission system from section 8.3.9.4 will be used to demonstrate the cost minimisation analysis. For this example, it will be assumed that failure in the second and third phases are twice and four times as costly, respectively, as failure in the first phase, since if the system fails in phases one or two the launch can be aborted with minimal consequences. If the system fails in the third phase of the mission, however, the rocket is lost and the consequences are far higher.

Table 8.11 – Cost of system failure during each phase of the phased mission.

Phase Number	Cost of system failure
1	1
2	2
3	4

Table 8.12 gives the one-off costs of reducing the probability of failure over the phased mission for each component by 1 to 8 percent.

Table 8.12 – One-off costs of reducing probability of component failure over the phased mission.

	Component		
% Reduction in Component Failure Probability over Phased Mission	**A**	**B**	**C**
1	0.01	0.01	0.001
2	0.04	0.02	0.008
3	0.09	0.03	0.027
4	0.16	0.04	0.064
5	0.25	0.08	0.125
6	0.36	0.14	0.216
7	0.49	0.22	0.343
8	0.64	0.30	0.512

Table 8.13 gives the direct costs of failure in the phased mission for each component at their current reliabilities and if they were improved by 1 to 8 percent. It shows that the direct cost of failure for components *A* and *C* increases if their reliability is improved, whilst it remains constant for component *B*.

Table 8.13 – Direct cost of failure for each component in the phased mission.

	Component		
% Reduction in Component Failure Probability over Phased Mission	**A**	**B**	**C**
0	0.01	1	0.01
1	0.01	1	0.02
2	0.01	1	0.04
3	0.08	1	0.08
4	0.08	1	0.16
5	0.08	1	0.32
6	0.2	1	0.64
7	0.2	1	1.28
8	0.2	1	2.56

Table 8.14 shows the change in the expected one-off improvement, direct component failure, mission failure and overall reliability costs over a phased mission for component *A*. From this table it is clear that, for component *A*, the maximum reduction in the overall reliability cost of a phased mission is achieved by reducing its mission unreliability by around 2%. Similar tables can be produced for components *B* and *C* but these are not shown here.

Table 8.14 – Change in the expected phased mission reliability costs for different percentage reductions in the phased mission failure probability of component *A*.

% Reduction in Failure Probability of Component A over Phased Mission	Change in expected one-off improvement costs	Change in expected direct component failure costs	Change in expected mission failure costs	Change in expected overall reliability costs
1	0.01000	-0.00006	-0.07000	-0.06006
2	0.04000	-0.00012	-.014000	-0.10012
3	0.09000	0.04064	-0.21000	-0.07936
4	0.16000	0.04015	-0.28000	-0.07984
5	0.25000	0.03968	-0.35000	-0.06032
6	0.36000	0.10700	-0.42000	0.04700
7	0.49000	0.10580	-0.49000	0.10580
8	0.64000	0.10460	-0.56000	0.18460

Figure 8.10 shows a plot of the change in the expected overall reliability cost of a phased mission against the percentage reduction in the phased mission failure probability for each component. This shows that the maximum expected cost reductions are achieved by reductions in the mission failure probability of around 2%, 5% and 3% for components *A*, *B* and *C* respectively. Note that this analysis and Equation 8.45 are based on the assumption that only the mission failure probability of a single component is reduced. If more than one component is to have its mission failure probability changed then the optimum percentages may vary due to the presence of higher order interactions between components.

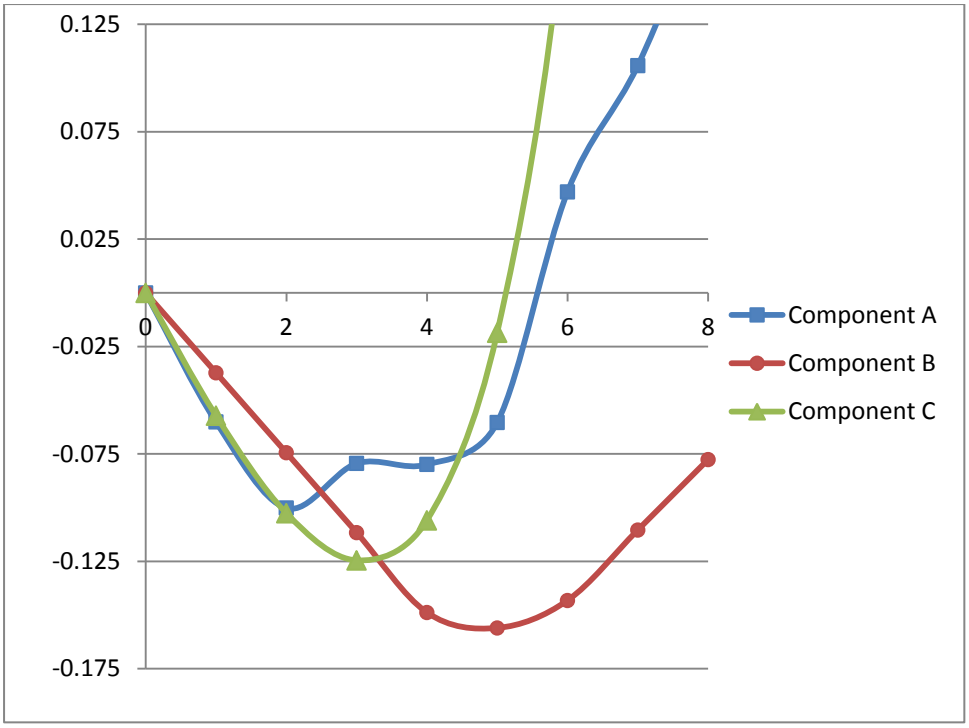


Figure 8.10 – Plot of the change in the expected overall reliability cost of a phased mission against percentage reduction in phased mission failure probability for each component.

8.8 Summary and Conclusions

The literature review found that phased mission interpretations had been developed for only two of the six commonly used non-phased mission system importance measures. Each importance measure is useful in certain situations and allows the aggregating of the importance values to give an average overall importance. Additionally, the two that had been extended do not share the same precise predictive qualities of their non-phased mission system counterparts. Phased mission extensions for all six of the commonly used importance measures were therefore developed and these provide the same use cases and useful properties as the non-phased mission system equivalents.

The existing phased mission importance measures were defined for the measurement of component importance in a certain phase or over the whole mission to the system reliability in a certain phase or over the whole mission. It was realised that the phased mission importance measures would be even more useful if they were also able to measure component importance from any period of the mission to the system reliability in any period of the mission, such as periods that do not start or end at the phase boundaries. An example of where this is useful is measuring the influence of infant mortalities for a component, where the infant mortality period does not coincide with a specific phase or set of phases, on the mission reliability. The new importance measure definitions were therefore developed so that they could be used to measure importance for very precise component and system reliability periods.

Another area for improvement of the existing phased mission importance measures was reducing the computational expense of finding the component importance to the system reliability over a number of phases, such as a complete phased mission. The new Phased Criticality II importance measure, introduced in this chapter, has better performance but still involves extensive calculations for missions with many phases. This led to the development of the Phased Criticality III importance measure that gives the same rankings as the Phased Criticality II importance measure but with much lower computational expense. This is therefore well suited to use with phased missions with large phase counts or when the speed of analysis is important such as in real time analysis.

The measurement of the importance of a group of components, rather than the individual importance of each, is essential in certain circumstances such as when investigating changes that will affect the reliability of multiple components in a system. However, whilst the literature review uncovered a selection of group importance measures for non-phased mission systems, none had been developed for phased mission systems. The group importance measures that were developed and presented in this chapter fill the gap in those available for phased mission systems and example scenarios of where they might be used are presented to demonstrate their practical use.

Finally, a common goal for those concerned with the reliability of a phased mission system is to minimise its reliability costs, where these costs include those incurred due to the consequences of mission failure. A method has therefore been developed that shows the precise amount by which the reliability of a component should be improved to minimise these costs. The use of this method eliminates guesswork from a reliability improvement exercise to show how the phased mission system reliability costs can be optimised.

In summary, the work presented in this chapter has established a set of methods for the optimisation of the reliability of a phased mission system that matches those available for non-phased mission systems. The availability of these means that it is now possible to approach the optimisation of the reliability and reliability costs of a phased mission system in the systematic and rigorous manner previously only possible for the non-phased case.

9 Conclusions and Areas for Further Work

The first part of this chapter contains a summary of the work and the conclusions that have been reached. The second part discusses some possible areas in which the work could be continued in the future.

9.1 Summary and Conclusions

It would be difficult to design phased mission systems to a desired reliability without methods capable of predicting the reliability of a particular system design from knowledge of its reliability structure and the reliability of its constituent components. For this reason, the development of these methods has received considerable research interest. The literature review covering methods developed for non-repairable phased mission systems showed that the earliest method, developed by Esary and Ziems [3], relied on transforming a phased mission into an equivalent non-phased mission. The method uses cut-set based techniques for the reliability evaluation and since the transformation process increases the number of basic events and therefore also the number of cut-sets, it results in significant computational expense. Later, researchers developed methods utilising specially developed algebras to deal with the dependencies between component failures in different phases thus removing the need for the transformation to a single phase. Amongst the research reviewed, was a method by Kohda et al [28] who formed a phased algebra intended for the analysis of accident sequences using event trees. A comparison between the various phased algebra based methods showed that all give correct results and that the method from Kohda et al results in the fewest terms in the final probability equation and lowest computational expense. The most recent analysis methods found in the literature make use of the BDD technique, the first of which was developed by Zang et al [31]. Its vastly better computational efficiency when deriving the exact reliability of a phased mission was discussed along with its limitation of only being able to model single failure mode components. Tang and Dugan [36] extended this method to model multiple failure modes by integrating it with a multiple failure mode non-phased mission BDD model developed by Zang et al [34]. Problems with this method were identified that cause it to give incorrect reliability results when applied to certain phased mission systems, as was shown through its application to some simple examples. This finding provided the motivation for the development of a correct method for the analysis of such systems. Additionally, it was decided that the method should be designed for high computational efficiency during repeat analysis of the same reliability structure but with component reliability model changes – as this is required for real time and importance measure analysis which were identified as important use cases. This led to the development of a BDD based method that uses a new intermediate step, where a dependency resolved data structure is constructed prior to the reliability evaluation, to improve repeat reliability analysis performance. It makes uses of a new set of BDD construction rules to form the BDD, including a new rule that reduces BDD size, together with the application of a new phased algebra derived from that of Kohda et al to form the pre-evaluation data structure. A substantial computer code has been created that implements this method, taking the phase fault trees and component models as inputs and outputting the probability of in-phase, transitional and overall mission failure as desired. It supports a number of common component failure models such as the exponential and Weibull distributions, and is easily extended to include further models. The code also includes an implementation of the method by Tang and Duggan and comparisons between the two in the analysis of a

variety of randomly generated phased missions showed that the new method results in smaller BDDs, faster repeat evaluations and much improved accuracy.

For repairable phased mission systems, the literature review showed that the earliest known method for the analysis of repairable phased mission systems is a Markov model based method developed by Alam and Al-Saggaf [44] that uses separate state transition matrixes for each phase. That method results in large state spaces even for relatively simple phased missions and to mitigate this problem Kim and Park [45] developed a method that reduces the state space by removing absorbing states from the transition matrices. The disadvantage of the absorbing state removal is that the method can only be used to find overall mission reliability and not that of individual phases, although it is sometimes only the former that is of interest to the analyst anyway. An alternative to the use of separate phase state transition matrixes is to use a single state transition matrix and phase varying transition rates, as in the method developed by Dugan [46]. This results in a state space that is the same as that from the method by Alam and Al-Saggaf. However, it has the disadvantages that phase by phase state space minimisation cannot be obtained and that numerical solution schemes, which are more computationally expensive than analytical solutions, must be used. Another Markov method, this time for the analysis of repairable phased missions with additional complications such as time dependant failure rates and random phase transition times has also been developed by Smotherman and Zemoudeh [47]. Despite the power of the method in dealing with many types of dependencies, it results in very large state spaces and therefore is only suitable for the analysis of relatively small systems compared to the other methods. The literature on the Markov based methods does not mention the computational expense of solving the transition matrices and this was found to increase as the cube of the matrix order, i.e. a matrix twice as big takes approximately eight times as long to compute, with modern computation techniques [49]. Due to the potential state space explosion problem associated with the Markov based methods, which can prevent large systems being analysed, Wang and Trivedi [48] developed a method that uses both Markov and BDD techniques. Although the computational expense of this method is much lower, it does make the assumption that a repaired component is only integrated back into the system at the end of the phase in which it is repaired. This means that it may not be accurately model cases where repaired components are integrated back into the system at the moment that repair completes. It is also only capable of modelling multiple failure modes at the component level, i.e. it cannot model the different effects of different failure modes on the system reliability. Monte Carlo simulation and Petri nets have also been used to analyse Phased Missions, such as the method by Chew et al [53] developed for modelling maintenance free operating periods. It was clear from the literature review that the major problem with the methods developed for repairable phased mission systems was that they had high computational expense due to the presence of repair dependencies that limits the size of the system they can analyse. The research therefore concentrated on how they could be made more efficient and this led to the development of three improvements to the existing methods. In addition, it was realised that although many real world systems contain components that can be repaired during a phased mission, often they represent only a small proportion of the total components in the system. This motivated the research into the development of methods specifically for the analysis of such systems that could deal with the repair dependencies, where present, whilst considerably reducing the computational expense. The first improvement was a simple extension to the Markov method by Kim and Park to the analysis of systems containing components with multiple failure modes, thus expanding its

application to a greater range of systems. The Kim and Park method requires the identification of the system failed states, for which they did not provide a method, and it was shown that even simple phased mission systems can have hundreds of thousands of system states. The second method developed was therefore an efficient method for performing this step with low computational expense by using the BDD technique in a novel way. The third method improved the efficiency of the evaluation scheme for the BDD based method from Wang and Trivedi as it avoids unnecessary repetition of probability evaluations when the same reliability structure is analysed more than once through the use of a new data structure. The fourth method developed is an efficient method for the analysis of systems containing both repairable and non-repairable components when repaired components are integrated back into the system at the end of the phase in which the repair completes. A simple example demonstrated that this method can result in significantly smaller BDDs than the method by Wang and Trivedi, over 60% smaller for the example given, if the system contains a large proportion of non-repairable components. It also has the advantage of using a BDD rather than Markov evaluation scheme for the non-repairable parts of the system. Together these two factors result in the much lower computational expense for the analysis of partially non-repairable phased mission systems, which was demonstrated for an example phased mission system. It also has the advantage of being able to model multiple failure modes at the system level for the non-repairable components thus allowing it to be used for the analysis of a greater range of system types. For the case of systems with component repairs that are integrated back into the system immediately upon completion, an efficient BDD based method was presented that gives an upper bound on the unreliability of a phased mission. The upper bound reliability condition was discussed and the factors affecting the accuracy of the estimate given, such as its increased accuracy if the non-repairable components have failure rates that do not increase with time. The method has the added benefit of being able to model multiple failure modes at the component and system levels for both non-repairable and repairable components. A software implementation of this latter method has been created and was demonstrated by applying it to the analysis of an example phased mission system. The discussion of this includes a comparison between the computational expense of the new method and the method by Kim and Park.

A thorough review of existing importance measures was carried out, uncovering that six for individual components and five for groups of components have been developed for non-phased mission systems compared to only two for individual components from phased mission systems. Each of the importance measures developed for non-phased missions is useful in a specific situation, for example the RRW importance measure [54] when the aim is to find the component with the possibility for increasing the system reliability by the greatest amount and the Criticality importance measure when the aim is to find the component with the greatest present influence on the system unreliability. This established the need for the development of a similar number of importance measures for the analysis of phased mission systems. It was also shown that those that do already exist lack some of the predictive qualities of their non-phased mission equivalents. Finally, the advantages of phased mission importance measures that are not restricted in the ‘cause’ and ‘effect’ time periods they measure were shown. Phased mission equivalents were therefore developed for each importance measure, including improvements to those already extended to phased missions. In total, seven new individual and six new group phased mission importance measures have been defined. They all have lesser restrictions on time periods compared to the existing phased mission importance measures and, where based on an existing non-phased

mission definition, have the same predictive qualities. For each new importance measures, the definition, the meaning of a specific importance value and the application to a realistic example scenario were explained. It was also shown that each of the new importance measure definitions retains the same interpretation as its non-phased mission counterpart. For example, it was shown that the value of the new Birnbaum importance measure extension gives the ratio of a change in the failure probability of the component to that of the system and that the value of the new Criticality importance measure extension gives the influence that the component has on the system reliability. The computational expense was compared for the different phased mission Criticality importance measure definitions, both new and that found in the literature, and shown to be significantly lower for the new definitions – particularly when used in the analysis of high phase count phased missions. The new group importance measures should prove useful as often reliability improvement efforts involve multiple components and prior to their development no importance measures for groups of components in a phased mission system existed. The computer code for the analysis of non-repairable phased missions has been extended to include importance measures such as the Phased RAW and Phased Criticality III so that they are quickly and easily calculated even for complex phased missions. Since the optimum system design, from a reliability perspective, is often that which has minimum reliability costs and not necessarily unreliability itself, a method for determining the component for which a reliability improvement, and the level by which it should be improved, to achieve the optimum decrease in expected reliability costs over a phased mission was presented. The application of this method to the optimisation of a simple example phased mission system demonstrated the ease with which it can be used to find the optimal path to reliability cost minimisation through the improvement of a component. Overall, the increase in the number, predictive qualities, flexibility with regard to time periods, ability to measure importance of groups and computational efficiency of the developed phased mission importance measures represent a significant improvement in the tools available for the optimisation of phased mission systems. Whereas previously it was non-phased mission systems that were easier to optimise through the use of importance measures, the status is now at least equal. The situation for phased mission systems could even be considered better if, for example, the ability to optimise system reliability in particular time periods, which is not possible with non-phased mission importance measures, is taken into account.

In conclusion, both of the objectives for the research have been met. The first research objective was to develop computationally efficient methods for the measurement of phase and mission reliability for both non-repairable and repairable systems. For non-repairable systems, specifically those with multiple component failure modes, the newly developed method has been shown to be more efficient than the existing alternatives. Comparatively smaller efficiency improvements were made to methods for systems containing only repairable components and some new techniques were introduced. However, the development of the first methods for those systems containing a mixture of repairable and non-repairable greatly improves the efficiency in the analysis of many real world phased mission systems that feature repair, since they often contain a majority of non-repairable components. The second research objective was to develop importance measures that can be used to guide phased mission system reliability improvement and optimisation actions. It has been shown that the set of newly developed phased mission importance measures are capable of identifying the areas of the system to target for a particular reliability improvement objective with greater precision and efficiently than was previously possible. They should prove useful to any engineer wishing to optimise the reliability of a phased mission system and this

research objective has therefore also been met. In addition to meeting the research objectives, a substantial computer code has been developed in parallel with the theoretical research and would provide a solid foundation on which to base any future implementation of the methods in industrial applications or as part of a commercial or open source software tool.

9.2 Areas for Further Work

Some areas for further work have been identified and are discussed in this section.

9.2.1 Analysis Methods

There are certain phenomena that can be modelled by certain methods found in the literature but not by the newly developed methods. These include common cause failures, uncovered failures and random phase transitions. The extension of the new methods to include them would be useful as they are sometimes present in real world systems and, in such cases, not including them in the modelling may lead to inaccurate results.

An obvious area for further work is the application of the methods to the analysis of an actual real world system. Due to a lack of access to data on real world systems, the phased missions that the new methods have been tested against has been limited to either simple or large but randomly generated example phased mission systems. The application of the methods to a real world system would serve to demonstrate how they could be applied by industry to help create reliable phased mission systems.

The methods introduced in chapter 6 are the first to have been specifically developed for systems containing both non-repairable and repairable components. Since many real world systems contain only a small proportion of components that are repairable during a phased mission and due to the computational overhead present in methods developed for repairable systems, this seems to be an interesting area for further study.

9.2.2 Computer Code

The computer code that has been developed, implementing many of the methods presented in this thesis, provides an analysis engine for phased mission analysis. This gives the potential for it to be used in a variety of ways, for example embedded within a hardware system or as part of a desktop application. It would be useful to develop a graphical user interface (GUI) that provides an easy to use means of interacting with this analysis engine and presents its output in an attractive and descriptive manner. Another, more specific, potential application that could be developed is a computer code for an unmanned aerial vehicle (UAV) that utilises real time analysis provided by the engine in mission reconfiguration decisions. For example, it could feed the analysis engine data about its current system status, such as known failed components, mission status and potential configurations to determine predicted reliabilities and then use this information to decide how to continue. A final example of an application that could be developed on top of the analysis engine would be an automated phased mission system reliability optimiser that takes inputs of the system reliability structure, component reliabilities, reliability costs and desired reliability goal (for example, 10% reduction in phase 2 failures and 15% reduction in mission failures) and determines, through the calculation of component importance and other analysis, the best path to achieving that goal.

Due to technological limitations, the historical trend that the available CPU speed doubles every two years (sometimes referred to as Moore's law), has not continued in recent years. The recent trend has been toward increasing the number of CPUs on a single computer and the use of distributed computing, where multiple computers connected across a network are used for a single task, rather than increasing the CPU speed, which has remained relatively constant since 2003 [63]. However, unlike increases in CPU speed, increases in the number of CPUs will not lead to a significant improvement in the performance of a sequential computer code that executes one line of code at a time. It is therefore necessary to parallelise the code so that multiple instructions are performed concurrently and all the available processing power across multiple CPUs is put to use. Based on this, an interesting area for future work would be to investigate ways in which algorithms for the methods in this thesis or new methods could be developed to profit from this trend.

References

[1] X. Dazhi and W. Xiaozhong, "A practical approach for phased mission analysis," *Reliability Engineering & System Safety*, vol. 25, pp. 333-347, 1989.

[2] J. D. Andrews, "A ternary decision diagram method to calculate the component contributions to the failure of systems undergoing phased missions," *Journal of Risk and Reliability*, vol. 222, pp. 173, 2008.

[3] J. D. Esary and H. Ziehms, "Reliability of Phased Missions," *Reliability and Fault Tree Analysis, Society for Industrial Applied Mathematics*, pp. 213-236, 1975.

[4] United States Department of Defense, "Definitions of terms for Reliability and Maintainability," *MIL-STD-721C*, 1981.

[5] D. R. Marples, "Chernobyl': five years later," *Soviet Geography*, vol. 32, pp. 291-313, 1991.

[6] S. Blake, P. Taylor, P. Tavner and G. Howarth, "An investigation into using oil replenishment data as a factor to prioritise the replacement of extra high voltage electrical cable," in 2009, pp. 30.

[7] J. Li, S. Pollard, G. Kendall, E. Soane and G. Davies, "Optimising risk reduction: An expected utility approach for marginal risk reduction during regulatory decision making," *Reliab. Eng. Syst. Saf.*, vol. In Press, Corrected Proof, .

[8] R. E. Melchers, "On the ALARP approach to risk management," *Reliab. Eng. Syst. Saf.*, vol. 71, pp. 201-208, 2, 2001.

[9] J. D. Andrews and S. J. Dunnett, "Event-tree analysis using binary decision diagrams," *Reliability, IEEE Transactions on*, vol. 49, pp. 230-238, Jun, 2000.

[10] P. D. T. O'Connor, *Practical Reliability Engineering, Fourth Edition.* Wiley, 2002.

[11] W. E. Vesely, F. F. Goldberg, N. H. Roberts and D. F. Haasl, *Fault Tree Handbook.* Washington, D.C.: U.S. Nuclear Regulatory Commission, 1981.

[12] J. D. Andrews, "The use of not logic in fault tree analysis," *Qual. Reliab. Eng. Int.*, vol. 17, pp. 143-150, 2001.

[13] J. D. Andrews and T. R. Moss, *Reliability and Risk Assessment.* Professional Engineering Publishing Limited, 2002.

[14] A. Rauzy, "New algorithms for fault trees analysis," *Reliability Engineering & System Safety*, vol. 40, pp. 203-211, 1993.

[15] R. Bryant, "Graph based algorithms for Boolean function manipulation," *IEEE Trans. Computers*, vol. 35, pp. 677-691, 1986.

[16] C. E. Shannon, "A symbolic Analysis of relay and switching circuits," *Transactions of the AIEE*, vol. 57, pp. 713, 1938.

[17] B. Bollig and I. Wegener, "Improving the variable ordering of OBDDs is NP-complete," *Computers, IEEE Transactions on*, vol. 45, pp. 993-1002, 1996.

[18] L. M. Bartlett and J. D. Andrews, "Efficient basic event ordering schemes for fault tree analysis," *Quality and Reliability Engineering International*, vol. 15, pp. 95, 1999.

[19] E. Hairer, C. Lubich and M. Roche, *The Numerical Solution of Differential Algebraic Systems by Runge-Kutta Methods.* Springer, 1989.

[20] H. C. Tijms, *A First Course in Stochastic Models.* Wiley, 2003.

[21] W. G. Schneeweiss, *Petri Nets for Reliability Modeling.* LiLoLe-Verlag GmbH, 1999.

[22] K. Jensen, "A brief introduction to coloured Petri Nets," *Tools and Algorithms for the Construction and Analysis of Systems,* pp. 203-208, 1997.

[23] M. Marseguerra and E. Zio, *Basics of the Monte Carlo Method with Application to Systems Reliability.* LiLoLe-Verlag GmbH, 2002.

[24] J. Andrews, "Identifying the major contributions to risk in phased missions," in *Reliability and Maintainability Symposium, 2006. RAMS '06. Annual,* 2006, pp. 624-629.

[25] K. B. Misra, Ed., *Handbook of Performability Engineering.* Springer London, 2008.

[26] G. R. Burdick, J. B. Fussell, D. M. Rasmuson and J. R. Wilson, "Phased Mission Analysis: A Review of New Developments and an Application," *IEEE Transactions on Reliability,* vol. R-26, pp. 43-49, 1977.

[27] H. Ziehms, "Reliability Analysis of Phased Missions," *Thesis, Naval Postgraduate School, Monterey, CA,* 1974.

[28] T. Kohda, M. Wada and K. Inoue, "A simple method for phased mission analysis," *Reliability Engineering & System Safety,* vol. 45, pp. 299-309, 1994.

[29] D. M. Rasmuson, "A comparison of the small and large event tree approaches used in PRAs," *Reliab. Eng. Syst. Saf.,* vol. 37, pp. 79-90, 1992.

[30] A. K. Somani and K. S. Trivedi, "Boolean Algebraic Methods for Phased-Mission System Analysis," *Proceedings of Sigmetrics,* pp. 98-107, 1994.

[31] Z. Zang, H. Sun and K. S. Trivedi, "A BDD-Based Algorithm for Reliability Analysis of Phased-Mission Systems," *IEEE Transactions on Reliability,* vol. 48, pp. 50, March 1999, 1999.

[32] R. La Band and J. D. Andrews, "Phased Mission Modelling using Fault Tree Analysis," *Reliability Engineering and System Safety,* vol. 78, pp. 45-56, 2002.

[33] R. La Band, "Systems Reliability for Phased Missions," *Thesis, Loughborough University, Leics, UK,* 2005.

[34] Z. Zang, D. Wang, H. Sun and K. S. Trivedi, "A BDD-Based Algorithm for Analysis of Multistate Systems with Multistate Components," *IEEE Trans. Computers,* vol. 52, pp. 1608, December 2003, 2003.

[35] L. Caldarola, "Coherent systems with multistate components," *Nuclear Engineering and Design,* vol. 58, pp. 127-139, 5, 1980.

[36] Zhihua Tang and J. B. Dugan, "BDD-based reliability analysis of phased-mission systems with multimode failures," *Reliability, IEEE Transactions on,* vol. 55, pp. 350-360, 2006.

[37] Huitian Lu, W. J. Kolarik and S. S. Lu, "Real-time performance reliability prediction," *Reliability, IEEE Transactions on,* vol. 50, pp. 353-357, Dec, 2001.

[38] D. Liu, R. Wasson and D. Vincenzi, "Effects of System Automation Management Strategies and Multi-mission Operator-to-vehicle Ratio on Operator Performance in UAV Systems," *Journal of Intelligent and Robotic Systems,* vol. 54, pp. 795-810, 05/01, 2009.

[39] Y. Mo, "Variable Ordering to Improve BDD Analysis of Phased-Mission Systems With Multimode Failures," *Reliability, IEEE Transactions on,* vol. 58, pp. 53-57, March, 2009.

[40] B. Evien, K. Sharkey, T. Thangarathinam, M. Kay, A. Vernet and S. Ferguson, *Professional XML.* Wrox, 2007.

[41] A. Hejlsberg, M. Torgersen, S. Wiltamuth and P. Golde, *C# Programming Language: The Annotated Edition.* Addison Wesley, 2008.

[42] B. Stroustrup, *The C++ Programming Language.* Addison Wesley, 1997.

[43] B. Eckel, *Thinking in C++.* Prentice Hall, 2000.

[44] M. Alam and U. M. Al-Saggaf, "Quantitative Reliability Evaluation of Repairable Phased-Mission Systems Using Markov Approach," *IEEE Transactions on Reliability,* vol. 35, pp. 498, 1986.

[45] K. Kim and K. S. Park, "Phased-Mission System Reliability under Markov Environment," *IEEE Transactions on Reliability,* vol. 43, pp. 301, 1994 June, 1994.

[46] J. B. Dugan, "Automated Analysis of Phased-Mission Reliability," *IEEE Transactions on Reliability,* vol. 40, pp. 45, 1991.

[47] M. K. Smotherman and K. Zemoudeh, "A Non-homogeneous Markov Model for Phased-Mission Reliability Analysis," *IEEE Transactions on Reliability,* vol. 38, pp. 585, 1989.

[48] D. Wang and K. S. Trivedi, "Reliability Analysis of Phased-Mission System With Independent Component Repairs," *IEEE Transactions on Reliability,* vol. 56, pp. 540, September 2007.

[49] C. Moler and C. Van Loan, "Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later," *SIAM Rev,* vol. 45, pp. 3, 03, 2003.

[50] A. Rauzy, "An experimental study on iterative methods to compute transient solutions of large Markov models," *Reliab. Eng. Syst. Saf.,* vol. 86, pp. 105, 2004.

[51] T. C. Sharma and I. Bazovsky Sr., "Reliability analysis of large system by Markov techniques," *Reliability and Maintainability Symposium, 1993. Proceedings. , Annual,* pp. 260-267, 1993.

[52] R. E. Altschul and P. M. Nagel, "The efficient simulation of phased fault trees " *Proceedings Annual Reliability and Maintainability Symposium,* pp. 292, 1987.

[53] S. P. Chew, S. J. Dunnett and J. D. Andrews, "Phased mission modelling of systems with maintenance-free operating periods using simulated Petri nets," *Reliability Engineering & System Safety,* vol. 93, pp. 980-994, 7, 2008.

[54] M. C. Cheok, G. W. Parry and R. R. Sherry, "Use of importance measures in risk-informed regulatory applications," *Reliability Engineering & System Safety,* vol. 60, pp. 213-226, 6, 1998.

[55] Z. W. Birnbaum, "On the importance of Different Components in a Multicomponent System," *Multivariate Analysis II,* pp. 581, 1969.

[56] Y. Dutuit and A. Rauzy, "Efficient algorithms to assess component and gate importance in fault tree analysis," *Reliability Engineering & System Safety,* vol. 72, pp. 213-222, 5, 2001.

[57] E. Borgonovo and G. E. Apostolakis, "A new importance measure for risk-informed decision making," *Reliability Engineering & System Safety,* vol. 72, pp. 193-212, 5, 2001.

[58] E. Borgonovo, "Differential, criticality and Birnbaum importance measures: An application to basic event, groups and SSCs in event trees and binary decision diagrams," *Reliability Engineering & System Safety,* vol. 92, pp. 1458-1467, 10, 2007.

[59] Jung Sik Hong and Chang Hoon Lie, "Joint reliability-importance of two edges in an undirected network," *Reliability, IEEE Transactions on,* vol. 42, pp. 17-23, 33, 1993.

[60] E. Zio and L. Podofillini, "Accounting for components interactions in the differential importance measure," *Reliability Engineering & System Safety,* vol. 91, pp. 1163-1174, 0, 2006.

[61] X. Gao, L. Cui and J. Li, "Analysis for joint importance of components in a coherent system," *European Journal of Operational Research,* vol. 182, pp. 282-299, 10/1, 2007.

[62] E. J. Henley and T. Inagaki, "Probabilistic evaluation of prime implicants and top events for non-coherent systems," *IEEE Transactions on Reliability,* vol. R-29, 1980.

[63] H. Stutter and J. Larus, "Software and the Concurrency Revolution," *ACM Queue,* 2005.