



The
University
Of
Sheffield.

Access to Electronic Thesis

Author: Alejandro Carlos Torres-Echeverria

Thesis title: Modelling and Optimization of Safety Instrumented Systems Based on Dependability and Cost Measures.

Qualification: PhD

Date awarded: 01/06/2009

This electronic thesis is protected by the Copyright, Designs and Patents Act 1988. No reproduction is permitted without consent of the author. It is also protected by the Creative Commons Licence allowing Attributions-Non-commercial-No derivatives.

This thesis was embargoed until 01/06/2010

Modelling and optimization of Safety Instrumented Systems based on dependability and cost measures

Alejandro Carlos Torres-Echeverría



Thesis submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

Department of Automatic Control and Systems Engineering
The University of Sheffield

May 2009

SUMMARY

This thesis is centred on modelling and multi-objective optimization of Safety Instrumented Systems (SIS) in compliance with the standard IEC 61508. SIS are in charge of monitoring that the operating conditions of a plant remain under safe limits and free of hazards. Their performance is, therefore, critical for the integrity of people around the plant, the environment, assets and production.

A large part of this work is devoted to modelling of SIS. Safety integrity and reliability measures, used as optimization objectives, are quantified by the Average Probability of Failure on Demand (PFD_{avg}) and the Spurious Trip Rate (STR). The third objective is the Lifecycle Cost (LCC); ensuring system cost-effectiveness. The optimization strategies include design and testing policies. This encompasses optimization of design by redundancy and reliability allocation, use of diverse redundancy, inclusion of MooN voting systems and optimization of testing frequency and strategies.

The project implements truly multi-objective optimization using Genetic Algorithms. A comprehensive analysis is presented and diverse applications to optimization of SIS are developed. Graphical techniques for presentation of results that aid the analysis are also presented.

A practical approach is intended. The modelling and optimization algorithms include the level of modelling detail and meet the requirements of IEC 61508. The focus is on systems working in low-demand mode. It is largely based on the requirements of the process industry but applicable to a wide range of other process.

Novel contributions include a model for quantification of time-dependent Probability of Failure on Demand; an approximation for STR; implementation of modelling by Fault Trees with flexibility for evaluation of multiple solutions; and the integration of system modelling with optimization by Genetic Algorithms. Thus, this work intends to widen the state-of-the-art in modelling of Probability of Failure on Demand, Spurious Trip Rate and solution of multi-optimization of design and testing of safety systems with Genetic Algorithms.

Statement of originality

Unless otherwise stated in the text, the work described in this thesis has been carried out solely by the candidate. None of this work has already been accepted for any other degree, nor is it being concurrently submitted in candidature for any degree.

Candidate: _____
Alejandro C. Torres-Echeverría

Supervisor: _____
Prof. Haydn Thompson

Acknowledgements

I would like to thank my supervisor Prof. Haydn Thompson for his guidance, support and patience through the development of my research project. He has certainly been a fundamental contributor to the completion of my PhD.

A very special mention and huge thanks to Prof. Sebastián Martorell from the Polytechnic University of Valencia, Spain. He provided a helpful and generous hand in difficult times, and enriched decisively my knowledge and contributed to my self-confidence in the topics exposed in this thesis, and thus my formation as researcher. I enjoyed very much the vivid discussions and exchange of ideas with him, which led to the publication of several joint articles.

I would also like to thank my friend Dr. Salem Adra for his guidance with Genetic Algorithms and his friendship.

This has been a large and eventful journey, which I have shared with many people that certainly made it enjoyable. To mention everyone here would be an impossible task, and I thank them all. However, I should make a special mention for my friend Dr. Miguel Gamma with whom I went through this life-changing journey, growing together and forming an irreplaceable friendship in the pathway. My life has also been enriched by my friends Jandia Martínez, Hector Barron and Guillermo Valencia, for what I am thankful and indebted. My friend of many years Juan Adriano has also somehow shared this journey with me. It has been stimulating to have a good friend that since the years of high school have been following parallel paths even without intention, thus growing together. He also provided an enormous uninterested helping hand in difficult financial times; helping especially to make possible those fabulous travels to different parts of the world! I would also like to mention my friend and former housemate Raquel Llorente, who was my accomplice during my first year at Sheffield and taught me to love everything Spanish.

My beloved Jarus has been a supportive and loving partner and my shining sun during this journey. She has made my stay in Sheffield beautiful, and changed my life filling it with love and tenderness. Thanks to you love.

I would also like to thank my beloved mother and my entire family, who are always in my thoughts wherever I go.

Finally, I want to acknowledge the support of the Mexican Council for Science and Technology, CONACyT, that funded this work making it thereby possible.

To my beloved Jarus...
For filling my life with love and tenderness

CONTENTS

SUMMARY	ii
Acknowledgements	iv
List of acronyms and abbreviations	xii
Nomenclature	xv
INTRODUCTION	1
1. OVERVIEW	1
2. MOTIVATION	1
3. STATEMENT OF THE PROBLEM	2
4. OBJECTIVES	3
5. MAIN NOVEL CONTRIBUTIONS	4
6. PUBLICATIONS	4
7. THESIS OUTLINE	5
CHAPTER 1	
SAFETY INSTRUMENTED SYSTEMS AND RAMS+C	8
MODELLING	
1.1. SAFETY INSTRUMENTED SYSTEMS	8
1.2. HARDWARE FAULT TOLERANCE	10
1.3. ARCHITECTURES FOR SIS	13
1.3.1. Voting architectures	13
1.3.2. Technologies used for SIS	15
1.3.3. Energized and de-energized systems	16
1.4. RELIABILITY AND SAFETY CONCEPTS	16
1.4.1. Reliability	18
1.4.2. Failure rate	18
1.4.3. Availability	19
1.4.4. System reliability	19
1.4.5. Diagnostic coverage	21
1.4.6. Common Cause Failure	21
1.4.7. Failure classification	21
1.4.8. Random hardware failure modes	22
1.4.9. Risk reduction	23
1.4.10. Probability of Failure on Demand	24

1.4.11. Spurious Trip Rate	25
1.5. REQUIREMENTS OF THE STANDARD IEC 61508	26
1.6. DEPENDABILITY MODELLING	29
1.6.1. Modelling methods	29
1.6.2. Overview of modelling of PFD for SIL analysis	31
1.6.3. Overview of modelling of STR for quantification of safe failures	33
1.6.4. Fault Tree Analysis	34
1.7. LIFECYCLE COST MODELLING	36
1.8. SAFETY INSTRUMENTED SYSTEM OPTIMIZATION ISSUES	39
CHAPTER 2	
RAMS+C OPTIMIZATION AND GENETIC ALGORITHMS	41
2.1. THE MULTI-OBJECTIVE OPTIMIZATION PROBLEM	41
2.1.1. The general problem	41
2.1.2. The multi-objective optimization problem for Safety Instrumented Systems	43
2.1.3. Pareto dominance and optimality	44
2.2. MULTI-OBJECTIVE OPTIMIZATION TECHNIQUES	46
2.2.1. Treating the multi-objective problem as a single-objective problem	46
2.2.2. Multi-objective optimization	47
2.3. OPTIMIZATION OF RAMS+C WITH GENETIC ALGORITHMS	50
2.4. PRINCIPLES OF GENETIC ALGORITHMS	60
2.4.1. Working principle	60
2.4.2. Development of Genetic Algorithms	62
2.4.3. The generic Genetic Algorithm	63
2.4.3.1. Initial population	63
2.4.3.2. Evaluation (fitness allocation)	64
2.4.3.3. Selection (for variation)	65
2.4.3.4. Crossover	67
2.4.3.5. Mutation	69
2.4.3.6. Reinsertion	71
2.4.3.7. Termination criteria	72
2.4.4. Multi-objective Genetic Algorithms performance questions	72
2.4.4.1. Exploration vs exploitation	72
2.4.4.2. Proximity (convergence)	72
2.4.4.3. Diversity	73
2.5. CONCLUDING REMARKS	73
CHAPTER 3	
OPTIMIZATION OF SIS DESIGN WITH PARALLEL REDUNDANCY	75
3.1. OPTIMIZATION OF SAFETY SYSTEM'S SPECIFICATIONS	75
3.2. FAULT TREES WITH HOUSE EVENTS	76
3.3. THE LIFECYCLE COST MODEL	77
3.4. FONSECA & FLEMING MULTI-OBJECTIVE GENETIC ALGORITHM	80
3.4.1. Pareto-based ranking	80

3.4.2. Fitness allocation	81
3.4.3. Presentation of results	83
3.5. DESCRIPTION OF THE APPLICATION PROBLEM	83
3.6. MODELLING AND QUANTIFICATION	87
3.7. IMPLEMENTATION OF THE OPTIMIZATION ALGORITHM	89
3.8. DISCUSSION OF RESULTS	91
3.9. CONCLUDING REMARKS	99
CHAPTER 4	
OPTIMIZATION OF DESIGN WITH DIVERSE REDUNDANCY	102
4.1. COMMON CAUSE FAILURE	102
4.1.1. The phenomenon of CCF	102
4.1.2. CCF modelling	103
4.1.3. Quantification of CCF at system level	105
4.1.4. Optimization considering CCF	108
4.2. DIVERSITY	109
4.2.1. The role of diversity against CCF	109
4.2.2. Diversity quantification	110
4.3. DESCRIPTION OF THE APPLICATION PROBLEM	111
4.4. QUANTIFICATION OF DEPENDABILITY OBJECTIVES	114
4.4.1. Fault Tree Analysis	114
4.4.2. Computer code for solving fault trees	116
4.5. THE LIFECYCLE COST MODEL	118
4.6. IMPLEMENTATION OF THE OPTIMIZATION ALGORITHM	119
4.7. DISCUSSION OF RESULTS	121
4.7.1. Architectures and their performance	121
4.7.2. Discussion of diversity	124
4.8. CONCLUDING REMARKS	128
CHAPTER 5	
MODELLING AND OPTIMIZATION OF PROOF TESTING POLICIES	130
5.1. OVERVIEW OF TESTING MODELLING AND OPTIMIZATION	130
5.1.1. Testing modelling	130
5.1.2. Testing optimization with Genetic Algorithms	133
5.1.3. Consideration of testing adverse effects	134
5.2. PROOF TESTING PRACTICES IN THE PROCESS INDUSTRY	135
5.3. TESTING BASIC CONCEPTS	137
5.3.1. Mean test cycle	137
5.3.2. Test Strategies	138

5.4. PFD TIME DEPENDENT MODEL	139
5.4.1. The PFD(t) baseline model	139
5.4.2. Inclusion of the effect of automatic diagnostics	140
5.4.3. The contribution of independent failures	141
5.4.4. The contribution of Common Cause Failure	142
5.4.5. Quantification of the average PFD	146
5.5. APPLICATION EXAMPLE	146
5.5.1. Influence of the test strategy	147
5.5.2. Influence of CCF	148
5.5.3. Influence of the diagnostic coverage	150
5.6. THE MULTI-OBJECTIVE PROOF TESTING POLICY PROBLEM	151
5.7. MODULARIZATION OF FAULT TREES WITH THE PFD(T) MODEL	152
5.8. SPURIOUS TRIP RATE MODEL	154
5.9. ELITIST NON-DOMINATED SORTING GENETIC ALGORITHM II	155
5.9.1. Non-domination sorting for ranking	156
5.9.2. Crowding distance density estimation	156
5.9.3. Crowded-comparison operator	157
5.9.4. The complete algorithm	157
5.9.5. Controlled elitism	159
5.10. APPLICATION CASE	160
5.10.1. Description of the problem and approach	160
5.10.2. Problem modelling	162
5.10.3. Implementation of the Genetic Algorithm	163
5.11. DISCUSSION OF RESULTS	165
5.12. CONCLUDING REMARKS	169
CHAPTER 6	
MODELLING AND OPTIMIZATION OF SIS INCLUDING MooN VOTING ARCHITECTURES	171
6.1. OVERVIEW OF MOON VOTING ARCHITECTURES	171
6.1.1. Effects of introducing voting architectures	172
6.1.2. Modelling of MooN architectures	173
6.1.3. Optimization with MooN systems	175
6.2. MODELLING PFD FOR MOON ARCHITECTURES	176
6.2.1. Previous considerations	176
6.2.2. Bypassing philosophy during test	177
6.2.3. Reduction of fault trees	180
6.2.4. CCF during test	181
6.2.5. Reconfiguration of fault trees with bypasses	182
6.2.6. Estimation based on first-order cut sets	185
6.2.7. Estimation of PFD considering null CCF	188
6.2.8. Application to the PFD(t) model	189
6.2.9. Application example	193
6.3. MODELLING STR FOR MOON ARCHITECTURES	194
6.3.1. Effects of bypass on the STR	194
6.3.2. Reconfiguration of STR fault trees with bypasses	194
6.3.3. Probability of safe failures	198

6.3.4. Quantification of STR modified during test	199
6.3.5. Test-induced STR	200
6.3.6. Application example	204
6.4. APPLICATION TO OPTIMIZATION OF SYSTEM DESIGN	205
6.4.1. Description of the problem	205
6.4.2. Implementation of the solution	206
6.4.3. Discussion of results	207
6.5. APPLICATION TO OPTIMIZATION OF TESTING POLICIES	212
6.5.1. Description of the problem	212
6.5.2. Implementation of the solution	213
6.5.3. Discussion of results	213
6.6. CONCLUDING REMARKS	219
CHAPTER 7	
CONCLUDING REMARKS	222
7.1. SCOPE OF THE THESIS	222
7.2. OPTIMIZATION DESIGN WITH PARALLEL REDUNDANCIES	223
7.3. OPTIMIZATION OF DESIGN WITH DIVERSE REDUNDANCY	225
7.4. OPTIMIZATION OF TESTING POLICIES	226
7.5. OPTIMIZATION OF DESIGN AND TEST WITH MOON VOTING ARCHITECTURES	228
7.6. IMPLEMENTATION OF THE GENETIC ALGORITHM	231
7.7. VISUALIZATION OF OPTIMAL SOLUTIONS	232
7.8. IMPORTANCE OF MULTI-OBJECTIVE OPTIMIZATION OF SIS AND APPLICABILITY OF THE METHODOLOGY	234
7.9. MAIN ACHIEVEMENTS OF THE THESIS	237
7.10. FUTURE WORK	238
REFERENCES	244
APPENDIX A	
EXTENDED TOPICS IN RELIABILITY AND SAFETY	254
A.1. DETAILED REVIEW OF DEPENDABILITY MODELLING	254
A.1.1. Modelling methods	254
A.1.2. Modelling PFD for SIL analysis	256
A.1.3. Modelling of STR	259
A.1.4. Fault Tree Analysis	261
A.2. CCF MODELLING	265
A.3. REVIEW OF MOON ARCHITECTURES	268
A.3.1. Definition of MooN architectures	268
A.3.2. Advanced voting techniques in NMR systems	270

A.3.3. Modelling of 1oo2D systems	271
-----------------------------------	-----

APPENDIX B	
ADVANCED TOPICS IN GENETIC ALGORITHMS	272

B.1. GENETIC DRIFT AND NICHING	272
---------------------------------------	------------

B.2. CONSTRAINT HANDLING	272
---------------------------------	------------

B.3. MULTI-CRITERIA DECISION-MAKING	273
--	------------

B.4. ADVANCED TOPICS IN THE FONSECA & FLEMING MOGA	274
---	------------

A.4.1. Sharing	274
----------------	-----

A.4.2. Niching by fitness sharing	275
-----------------------------------	-----

A.4.3. Mating restriction	275
---------------------------	-----

A.4.4. Articulation of preferences	276
------------------------------------	-----

B.5. DETAILED PROCEDURES IN THE NSGA-II	278
--	------------

A.5.1. Non-dominated sorting	278
------------------------------	-----

A.5.2. Crowding distance density estimation	278
---	-----

A.5.3. Controlled Elitism	279
---------------------------	-----

LIST OF ACRONIMS AND ABBREVIATIONS

ALARP	As Low As Reasonably Practicable
AOT	Allowed Outage Time
API	American Petroleum Institute
ARC	ARC Advisory Group
BDD	Binary Decision Diagram
BLX- α	Blend Crossover
BP	British Petroleum
CCF	Common Cause Failure
CCPS	Center for Chemical Process Safety
CMF	Common Mode Failure
CRelA	Continuous Reliability Allocation
CS	Cut Set
CSU	Critical Safety Unavailability
DDC	Dangerous Detected Common (cause failure)
DDN	Dangerous Detected Normal (independent failure)
DI	Diversity Index
DM	Decision Maker
DrelA	Discrete Reliability Allocation
DTU	Unavailability due to Test and Maintenance Downtime
DUC	Dangerous Undetected Common (cause failure)
DUN	Dangerous Undetected Normal (independent failure)
EA	Evolutionary Optimization
EIR	Extended Intermediate Recombination
ELR	Extended Line Recombination
ESD	Emergency Shutdown System
F&G	Fire and Gas detection system
FC	Final Control element
FCT	Fault Contribution Tree
FDS	Firewater Deluge System
FSC	Formal Software Construction Limited
FT	Fault Tolerance
FTA	Fault Tree Analysis
GA	Genetic Algorithm
HID	Hazardous Installations Directory
HIPS	High Integrity Protection System
HPIS	High Pressure Injection System
HSE	Health and Safety Executive
I/O	Input/Output
IEC	International Electrotechnical Commission
IP	International Practice
ISA	The Instrumentation, Systems and Automation Society
k -out-of- n :F	An n -component system that fails when k components are Faulty
k -out-of- n :G	An n -component system that success when k components are Good
LAC	Life Acquisition Cost
LCC	Lifecycle Cost
LS	Logic Solver
LSC	Life Support Cost
LUC	Life Unavailability Cost
MA	Markov Analysis
MATLAB [®]	MATrix LABoratory (programming language)
MCDM	Multi-Criteria Decision Making
MCS	Minimal Cut Set
MDT	Mean Down Time
MGL	Multiple Greek Letter (model)
MO	Multi-Objective
MOGA	Multi-Objective Genetic Algorithm (Fonseca & Fleming's)
MooN	M-out-of-N redundant arrangement

MooND	M-out-of-N with enhanced Diagnostics
MTBF	Mean Time Between Failures
MTTF	Mean Time To Fail
MTTR	Mean Time to Restoration
NORSOK	The Competitive Standing of the Norwegian Offshore Sector (in Norwegian)
NPGA	Niched-Pareto Genetic Algorithm
NPP	Nuclear Power Plant
NSGA-II	Non-dominated Sorting Genetic Algorithm II
NTI	Norwegian Technology Standards Institution (in Norwegian)
NUREG	Standard of the US Nuclear Regulatory Commission
OLF	Standard of the Norwegian Oil Industry Association
PAES	Pareto Archived Evolution Strategy
PDS	Reliability of computer based systems (in Norwegian)
PES	Programmable Electronic System
PFD	Probability of Failure on Demand
PFD _{avg}	Average Probability of Failure on Demand
PFD _{max}	Maximum Probability of Failure on Demand
PLC	Programmable Logic Controller
PRA	Probabilistic Risk Analysis
PT	Pressure Transmitter
PTIF	Systematic Test-independent failures
PWR	Pressurized Water Reactor
RAMS+C	Reliability, Availability, Maintainability and Safety plus Cost
RBD	Reliability Block Diagram
RedA	Redundancy Allocation
RelA	Reliability Allocation
RP	Recommended Practice
RPIS	Reactor Protection Instrumentation System
RPS	Random Probability Shock (model)
RRF	Risk Reduction Factor
RWS	Roulette Wheel Selection
S&M	Surveillance and Maintenance
SBX	Simulated Binary Crossover
SD	Safe Detected
SDC	Safe Detected Common (cause failure)
SDN	Safe Detected Normal (independent failure)
SE	Simplified Equations
SFF	Safe Failure Fraction
SIL	Safety Integrity Level
SINTEF	The Foundation for Scientific and Industrial Research (in Norwegian)
SIS	Safety Instrumented System
SO	Single-Objective
Sol #	Solution number
SPEA2	Strength Pareto Evolutionary Algorithm 2
SRS	Stochastic Universal Selection
SSGA	Steady State Genetic Algorithm
SSPR	Stochastic Sampling with Partial Replacement
SSR	Stochastic Sampling with Replacement
STI	Surveillance Technical Specifications
STR	Spurious Trip Rate
SUC	Safe Undetected Common (cause failure)
SUN	Safe Undetected Normal (independent failure)
SUS	Stochastic Universal Sampling
SUT	Safe failures Undetected by diagnostics but detected by proof Test
T&M	Test and Maintenance
TI	Test Interval
TMR	Triple Modular Redundant
TS	Test Strategy
TSM	Technical Specifications and Maintenance
TT	Temperature Transmitter

VEGA Vector Evaluated Genetic Algorithm
VPF Value of Preventing a Fatality

NOMENCLATURE

$A(t)$	Availability (time dependent)
$C_{100(N-1)}$	Beta modification factor for N-1 components (M=1)
$C_a(x)$	Yearly cost of accidents
C_{ACC}	Cost of an accident
CCF_{AB}	CCF of components A and B
CCF_N	CCF on N components
C_{CM}, C_{CM}^i	Corrective maintenance cost, corrective maint. cost of i^{th} component
C_{cons}	Consumption cost per year
C_{design}	Design cost
C_{HAZARD}	Cost of hazard
C_{hr}	Repair hourly cost
C_{hs}	Cost of production per hour
C_{ht}	Test hourly cost
$C_{inst/comm}$	Installation & commissioning cost
C_i^{spares}	Spares cost per repair (%purchase cost/event) of i^{th} comp.
C_{MooN}	Beta modification factor for MooN architecture
C_N	Sum of all C_{MooN} factor with equal N (Eq. (6.3))
$Components_i$	Number of components of the i^{th} subsystem
C_{OP}	Operation cost
C_{PM}, C_{PM}^i	Preventive maintenance cost; preventive maint. cost of i^{th} component
C_{PROC}	Procurement cost
$C_{purchase}, C_i^{purchase}$	Acquisition cost, acquisition cost of the i^{th} component
C_{RISK}	Risk cost
C_{SD}	Cost per spurious shutdown
C_{SP}	Average cost of expenditure of spares per repair
$C_{Start-up}$	Cost of first start-up
C_{STR}	Cost of production loss by spurious trip rate
$C_{T\&CM}$	Cost of test and corrective maintenance
C_T, C_i^T	Proof test cost; proof test cost of i^{th} component
C_{test}	Constant value of PFD_{ind} during test
C_{TMooN}	C_{MooN} factor modified during test
c_u	Cost of outage time
D	Dangerous
DDC	Dangerous Detected Common (cause failure)
DDN	Dangerous Detected Normal (independent failure)
DI	Diversity Index
d, d_{ij}	Distance between two solutions ij (in the objective space)
d_i, d_m^i	Crowding distance of solution i; Crowding distance of the m^{th} objective
DTU	Unavailability due to Test and Maintenance Downtime
DUC	Dangerous Undetected Common (cause failure)
DUN	Dangerous Undetected Normal (independent failure)
F	Number of failed units
$F(ACC PFD_{avg})$	Accident frequency without the SIS per year
$f(r)$	Assignment function in the MOGA
$f(t)$	Probability of failure
$F(t)$	Unreliability (time dependent)
$f(x)$	Objective function vector
f_i	Fitness of the i^{th} individual
F_i, f_i	Non-dominated front i^{th}
f_j^{CM}	Repair frequency of i^{th} component
$f_k^{(i)}$	Objective function value of the i^{th} individual
f_k^{max}	Maximum objective value of the k^{th} objective
f_k^{min}	Minimum objective value of the k^{th} objective
Fr	Frequency of safe failure
FT	Fault tolerance
F_{TR}	Tolerable risk frequency
G	Generational gap

$\mathbf{g}(\mathbf{x})$	Vector of constraints
H_N	β modification factor for independent failures of a N-component system
i	i^{th} component being tested or last tested subscript (in the PFD(t) model)
i	Subsystem subscript
I_i	Importance measure for cut set i
j	Technology type subscript
K	Staggering factor
K	Total number of non-dominated fronts
l	Chromosome length
l, ll	Lower limit
LCC	Lifecycle Cost
$LCC_{T\&CM}$	Test and corrective maintenance related LCC
m	Factor used in mutation for real numbers (Eq. (2.13))
M, m	Points defining a parallelogram in the objective space
M_{ij}	Maintenance frequency of ij component
mod	Modulo operator
m_{shr}	Mutation range shrinking factor
n	Number of point evaluations (in Eq. (5.20))
n	Number of components in a redundant subsystem
n	Number of objectives
n	Degree of diversity in a system (CCF Boundary model)
nc_i	Niche count
n_i	Number of individuals permitted from the i^{th} front
n_p	Dominance count
N_s	Number of components of a subsystem
N_i	Number of components of the i^{th} subsystem
N_{ij}	Number of ij components
N_{pop}	Size of population
N_s	Number of subsystems
N_{tests}	Number of tests
N_{trips}	Number of spurious trips
P	Test-staggering multiplication factor
p	Member of a population
p_s	Probability factor used in mutation for real numbers (Eq. (2.13))
$P()$	Probability of failure
$P(A)$	Probability of event A
$P(C_i)$	Probability of cut set i occurring
\mathbf{P}, \mathbf{P}_0	Array of the parent population, Initial (parent) population
p_c	Crossover probability
PFD	Probability of failure on demand
PFD(t), PFD(t_i)	Time-dependent PFD; time-dependent PFD point evaluation
PFD_{avg}	Average Probability of failure on demand
PFD_{CCF}	PFD from CCF
PFD_{IND}	Total PFD by combination of independent failures
$PFD_{ind,i}(t)$	PFD(t) from independent failure of the i^{th} component
PFD_{max}	Maximum (peak) PFD(t)
PFD_{MooN}	Total PFD of a MooN system
$PFD_{TOT}(t)$	Total time-dependent PFD
p_{in}^n	n^{th} variable of the father chromosome
$p_i^{(t)}$	Number of individuals at generation t
p_m	Mutation probability
p_{mn}	n^{th} variable of the mother chromosome
p_{new}^l	i^{th} gene of the offspring chromosome number l
p_{new}^i	Mutated individual i^{th}
p_{new}^n	n^{th} variable of an offspring
P_{pop}	Fixed size of parent population
P_{r-trip}	Probability of test-induced spurious trip
P_{sf}	Probability of safe failure
$P_{sf}(CCF)$	Probability of safe CCF
$P_{sf}(SD)$	Probability of safe detected failure

Psf(SN)	Probability of safe normal (independent) failure
Psf(SU)	Probability of safe undetected failure
Psf(SUT)	Probability of safe failure undetected by diagnostics but detected by proof test
\mathbf{P}_t	Parent population at generation t
PVF	Factor by present value
q	Constant unavailability symbol; unreliability
\mathbf{Q}	Array of the offspring population
$Q(A)$	Probability of independent failure of component A
$Q(\text{CCF}), Q(\text{CCF}_N)$	Probability of CFF; probability of CFF of N components
$Q(f)$	Probability of failure
$Q(\text{Ind})$	Probability of independent failure of one single component
$Q(t)$	Unavailability (time dependent)
$Q_{k/n}$	Unavailability of k -out-of- n system
$Q_{\text{sys}}(q(t))$	Probability of system unavailability
\mathbf{Q}, \mathbf{Q}_0	Offspring population set; first offspring population
\mathbf{Q}_t	Offspring population at generation t
r	Reduction rate parameter for NSGA-II
r	Rank value assigned to an individual
r	Reliability
range_i	Mutation range
rank	Rank value of an individual
Redundancy_i	Redundancy level of the i^{th} subsystem
r_s	Number indication position in RWS and SUS
$r_u^{(t)}$	Number of individuals preferable to x_u at generation t
R	Discount rate
R	Random number (in crossover operation)
$R(t)$	Reliability (time dependent)
$R(x)$	Yearly risk contribution
R_f	Reduction factor (in Eq. (2.12))
R_i	Reliability of i^{th} component
R_{sys}	System reliability
Rt_N	Relative time under normal operation
Rt_{TR}	Relative time under test and (possible) repair
\mathbf{R}, \mathbf{R}_t	Combined population parents+offspring; same population at generation t
s	Relative fitness desired for the best individual
S	Safe
S, S_D	Score for undetected failures; for detected failures (IEC 61508 β factor method)
S	Number of successful units (in a redundant arrangement)
SD	Safe Detected
SDC	Safe Detected Common (cause failure)
SD_{LOSS}	Cost of production loss per hour
SDN	Safe Detected Normal (independent failure)
SD_{Time}	Restart time after shutdown
SFF	Safe failure fraction
$Sh(d)$	Sharing function
SIL	Safety integrity level
$S_{k/n}$	Probability of spurious operation of k -out-of- n system
S_p	Set of solutions dominated by individual p
STR	Spurious trip rate
STR_λ	STR_λ caused by internal failure
$STR_{\lambda, \text{IND}}$	STR_λ total contribution by combination of independent failures
$STR_{\lambda, \text{CCF}}$	STR_λ contribution by CCF
STR_N	STR during normal operation
STR_T	Total spurious trip rate
STR_{test}	Proof test-induced STR
STR_{TR}	STR during test and repair
SUC	Safe Undetected Common (cause failure)
SUN	Safe Undetected Normal (independent failure)
t	Time
T	Total period evaluation time (mission time)

<i>Technologies_i</i>	Number of technologies used in the i^{th} subsystem
T_i, T_{ij}	Test interval; test interval of the i^{th} component of the j^{th} type
T_p	Time to first test,
T_{P1}	Time to first test of the first component
T_{Pi}	Time to first test of the i^{th} component
T_{PN}	Time to first test of the N^{th} component
T_r	Repair time
TS	Test strategy
T_s	Start-up time
T_t	Test time
<i>Type</i>	Type of device as variable
<i>u, ul</i>	Upper limit
u	Objective vector of individual u
U_{SF}	Unavailability by safe failure
v	Objective vector of individual v
w	Test interval remainder parameter
w_{CCF}	Test interval remainder parameter for CCF
w_i	Test interval remainder of the i^{th} component
x, X	Decision variables vector; decision space
x_i	i^{th} individual
x_u	Individual with objective vector u
y, Y	Objective function vector; objective space
α	Parameter for the sharing function (Eq. (A.2))
α	Random number in the interval [-0.25, 1.25] (only in Eq. (2.9))
α_i	Factor used in mutation for real numbers (Eq. (2.13))
β	Random number in the interval [0, 1] (only in Eq. (2.8))
β	β factor for quantification of CCF
β^-	β factor modified for MooN architecture (same as β_{MooN})
β_{CM}	β factor cost modifier (P, D, I, C for purchase, design, installation, consumption)
β^D	β factor for detected failures
β^{DD}	β factor for dangerous detected failures
β^{DU}	β factor for dangerous undetected failures
β^{-DD}	β_{MooN} for dangerous detected failures
β^{-DD}_{N-1}	β factor for N-1 components for dangerous detected failure
β^{-DU}	β_{MooN} for for dangerous undetected failures
β^{-DU}_{N-1}	β factor for N-1 components for dangerous undetected failure
β_{MooN}	β factor modified for MooN architecture
β^{SD}	β factor for safe detected failures
β^{SU}	β factor for safe undetected failures
β^U	β factor for undetected failures
δ	Factor used in mutation for real numbers (Eq. (2.11))
Δ	Difference between point M and m
ε	Diagnostic coverage
$\varepsilon^D, \varepsilon^{\text{Danger}}$	Diagnostic coverage for dangerous failures
$\varepsilon^S, \varepsilon^{\text{Safe}}$	Diagnostic coverage for safe failures
ε^{ts}	Diagnostic coverage of proof test on safe failures
λ	Representation of a offspring population
λ	Failure rate
$\lambda^{\text{DDC}}_{\text{test}}$	DDC failure rate modified during test
$\lambda^{\text{Detected}}$	Detected failure rate
$\lambda^{\text{DUC}}_{\text{test}}$	DUC failure rate modified during test
λ_l	Lower bound failure rate
λ^{STi}	Test-induced spurious trip failure rate per component
λ^C	Common cause failure rate
$\lambda^D, \lambda^{\text{Danger}}$	Dangerous failure rate
$\lambda^{\text{DC}}_{N-1}$	Detected common cause for N-1 components
λ^{DD}	Dangerous detected failure rate
λ^{DDC}	Dangerous detected common cause failure rate
$\lambda^{\text{DDC}}_{N-1}$	Dangerous detected CCF rate for N-1 components

λ^{DDN}	Dangerous detected normal (independent) failure rate
λ^{DN}	Detected normal (independent)
λ^{DU}	Dangerous undetected failure rate
λ^{DUC}	Dangerous undetected common cause failure rate
$\lambda^{DUC_{N-1}}$	Dangerous undetected CCF rate for N-1 components
λ^{DUN}	Dangerous undetected normal (independent) failure rate
λ^N	Normal (independent) failure rate
$\lambda^S, \lambda^{Safe}$	Safe failure rate failure rate
λ^{SC}	Safe common cause failure rate failure rate
λ^{SD}	Safe detected failure rate
λ^{SDC}	Safe detected common cause failure rate failure rate
λ^{SDN}	Safe detected normal (independent) failure rate
λ^{SN}	Safe normal (independent) failure rate
λ^{SU}	Safe undetected failure rate
λ^{SUC}	Safe undetected common
λ^{SUN}	Safe undetected normal (independent)
λ^{SUTC}	Safe undetected by automatic diagnostics but detected by test (CCF) failure rate
λ^{SUTN}	Safe undetected by diagnostics but detected by test (independent) failure rate
$\lambda^T, \lambda^{Total}$	Total failure rate
λ_{u}	Upper bound failure rate
$\lambda^{Undetected}$	Undetected failure rate
μ	Representation of a parent population
ρ	Factor used in exponential assignment function in the MOGA
ρ_i	Remaining slots
σ_{mate}	Mating restriction parameter
σ_{share}	Fitness sharing parameter (niche size)
%PCE	Fraction of component purchase cost
%Safe	Percentage of safe mode failures over total

INTRODUCTION

1. OVERVIEW

This thesis specifically addresses the modelling and optimization of Safety Instrumented Systems (SIS) including design and testing policies. These are the most important aspects to consider during the realization phase of the system lifecycle. It comprises a series of studies into different aspects of SIS. The intention is to implement a practical approach that permits application to industrial systems, and therefore it takes into account the requirements of the international standard IEC 61508. This implies that dependability modelling must be performed to the level of detail required by this standard. Research into reliability optimization rarely approaches this level of detail, and therefore this work pays special attention to it. The optimization is made using multi-objective Genetic Algorithms. Several different aspects of the design are gradually approached and incorporated into the analysis.

2. MOTIVATION

Safety Instrumented Systems are used in a wide range of different applications. They are employed in order to reduce risk of hazards to acceptable levels. It is known that their failure to perform their intended function could result in loss of the assets of a company, widespread damage to the environment, harm to personnel and people around the facilities and even loss of life. It is therefore crucial to ensure their effective performance, which is basically achieved by an optimal design and operation. This requires careful balance of the system benefits against its costs, which must include the entire system lifecycle cost (not only its acquisition cost). The system can be very efficient in terms of responding to hazardous events, but it must not be so sensitive that it could create spurious activations. This affects the trustworthiness that the operator places upon the system, and even its lifecycle costs. Therefore, this becomes a problem of balancing safety and reliability, and these two in turn against cost. This is the problem addressed in this work.

A safety instrumented system must follow a safety lifecycle in order to ensure that the required dependability level is achieved and actually maintained during its entire operating lifetime. This is mainly devised and executed during the realization phase of the system: that is during its design. It is this phase that is the focus of this study, to aid the decision making process during system design.

The two main ways of ensuring the adequate system cost-effective dependability are to achieve both an optimal design and an optimal proof test policy for its operation. These are addressed here.

In order to give a practical approach to the study, it is based on relevant standards and current best practice. The international standard IEC 61508 "Functional safety of electrical/electronic/programmable electronic safety-related systems" (IEC, 1998-2005) aims to provide a means of ensuring that safety is effectively reached based on functionality of electrical, electronic or programmable electronic systems. Since IEC 61508 is a generic document, non-specific to any industry sector, it is relevant to a large range of different sectors. This is the main standard used as the reference in this work, although some other relevant standards and approaches are explored as necessary.

In order to be able to achieve the objective of finding optimal designs and testing policies two fundamental areas must be put together. First of all, an adequate comprehensive model with a good level of resolution is required. Secondly, a powerful optimization tool capable of handling the specific functions of the models and delivering the results is needed. This thesis thus focuses on both the modelling and optimization aspects of safety systems.

3. STATEMENT OF THE PROBLEM

The main problem statement being addressed by this thesis is: "The development of Safety Instrumented Systems with the aim of achieving optimal designs and testing policies at the conceptual design stage, based on the most important design objectives, using a practical approach founded on current standards, and addressing some of the key issues".

This study is largely focussed on optimization of safety, which is the requirement of IEC 61508. However, reliability and cost are fundamental measures that must also be addressed during the system realization. This makes achievement of well balanced, safe, reliable and cost-effective systems possible. Therefore, the project bases the system optimization on the following three main objectives:

- Average Probability of Failure on Demand (PFD_{avg}). This is a measure of loss of safety by unavailability of the system (by dangerous failures), and determines the SIL achieved.
- Spurious Trip Rate (STR). This addresses the issue of reliability, and has an important impact on the trustworthiness that the user places upon the system and the overall system life cycle cost since it results in loss of production.

- Life Cycle Cost (LCC). Complete costing of the overall system life cycle, comprising design, procurement and implementation, operation and risk costs.

The project aims to integrate a methodology that gradually addresses different issues that affect and enhance the overall final design, namely:

1. Reliability and redundancy allocation for parallel architectures.
2. Design with diversity as a countermeasure to Common Cause Failure and its effects.
3. Formulation of the most effective proof test policies.
4. System design with MooN voting architectures for a better balance of objectives.

Genetic Algorithms are one technique of evolutionary computation which mimics the natural selective process of evolution for solution of optimization problems. Different from other optimization methods, they present several advantages: They are very useful for solving complex, high dimensional, discrete, non-linear and discontinuous problems, with capacity to handle integer variables. GAs are able to deal with problems where the objective function is not explicit (such as those generated by fault tree analysis). In addition, they provide the decision maker a pool of good optimal solutions. Genetic Algorithms are therefore the tool used for solution of the optimization problems herein.

4. OBJECTIVES

To develop a comprehensive study of multi-objective optimisation of safety instrumented systems for the realization phase based on important dependability measures and lifecycle cost.

Particular objectives:

- To develop the optimization for safety systems in compliance with relevant standards and current best practices
- To incorporate safety, reliability and lifecycle cost into the design optimization.
- To integrate safety and reliability modelling techniques with lifecycle costing into optimization by multi-objective Genetic Algorithms.
- To explore the most relevant design and testing issues (e.g. Common Cause Failure (CCF), testing, voting) during the optimization.
- To provide an integrated tool for safety system analysis.

5. MAIN NOVEL CONTRIBUTIONS

- Application of multi-objective Genetic Algorithms to optimisation of safety systems considering the level of modelling detail required by IEC 61508.
- Development of an integrated methodology compliant with IEC 61508 and focussed on the needs of the process industry.
- Development of a time dependent PFD model for inclusion of test, CCF and diagnostic coverage.
- Development of a Spurious Trip Rate model including testing adverse effects.
- Inclusion of Common Cause Failure analysis and reduction by redundancy with diversity.
- Optimization of testing policies with an adequate level of modelling detail and based on current best practice.
- Development of a diversity index to measure diverse redundancy effects.
- Study of MooN system's reconfiguration during test dependant upon the operation philosophy.
- Application of Fonseca and Fleming MOGA (Fonseca & Fleming, 1993) and NSGA-II (Deb et al., 2000, 2002) for RAMS+C optimization (Martorell et al., 2005a).

6. PUBLICATIONS

Journal articles

- Torres-Echeverria A.C., Thompson H.A. *Multi-objective genetic algorithm for optimization of system safety and reliability based on IEC 61508 requirements: a practical approach*. Proceedings of the IMechE Part O: Journal of Risk and Reliability 2007, Vol 221(O3), pp.193-205.
- Torres-Echeverria A.C., Martorell S., Thompson H.A. *Design optimization of a safety instrumented system based on RAMS+C addressing IEC 61508 requirements and diverse redundancy*. Reliability Engineering and System Safety 2009;94(2):162-179.
- Torres-Echeverria A.C., Martorell S., Thompson H.A. *Modelling and optimization of proof testing policies for safety instrumented systems*. Reliability Engineering and System Safety 2008. Article in press. doi:10.1016/j.res.2008.09.006.
- Torres-Echeverria A.C., Martorell S., Thompson H.A. *Modelling and optimization of safety instrumented systems with MooN voting architectures*. In preparation, to be submitted to Reliability Engineering and System Safety.

Conference and symposium contributions

- Torres-Echeverria A.C. and Thompson H.A. *Multi-objective genetic algorithm for optimization of system safety and reliability based on IEC 61508 requirements: a practical approach*. Procs. 17th Advances in Risk and Reliability Technology Symposium AR²TS, Loughborough University, Loughborough, UK, April 2007.
- Torres-Echeverria A.C., Martorell S. and Thompson H.A. *Optimization of RAMS+C for safety instrumented system design with diverse redundancy*. In: Aven A & Vinnem JE (eds), Procs. European Safety and Reliability Conference ESREL '07. Stavanger, Norway June 2007.
- Torres-Echeverria A.C., Martorell S., Thompson H.A. *Modelling test strategies effects on the Probability of Failure on Demand for Safety Instrumented Systems*. In: Martorell S. (ed), Procs. European Safety and Reliability Conference ESREL '08. Valencia, Spain, September 2008.
- Torres-Echeverria A.C., Martorell S., Thompson H.A. *Optimization of proof testing policies for safety instrumented systems using multi-objective genetic algorithms*. In: Martorell S. (ed), Procs. European Safety and Reliability Conference ESREL '08. Valencia, Spain, September 2008.

7. THESIS OUTLINE

In general, the thesis firstly provides the theoretical basis and literature review behind the work (Chapters 1 and 2), it then presents the bulk of the research (Chapters 3 to 6), and finally presents the concluding remarks. Since the research work comprises a series of studies on different issues of safety instrumented systems, chapters 3 to 6 are practically self-contained units that have this structure: Introductory theory, exposition of the approach, methodology, modelling, optimization case and presentation and discussion of results.

The outline of the thesis is as follows:

Chapter 1 presents the basic theory of Safety Instrumented Systems and system modelling necessary to understand the material in the thesis. A brief introduction to hardware fault tolerance and reliability theory is given. The chapter then describes the most important requirements of the standard IEC 61508. The most relevant models for dependability, and then for lifecycle costing, are described and analyzed. Additional background material is provided in Appendix A. After this general overview, the most important issues related to design and testing of safety instrumented systems are discussed.

Chapter 2 introduces the multi-objective optimization problem, followed by a review of optimization techniques. It then makes an account of optimization of RAMS+C, in both design and testing, putting special emphasis on application using Genetic Algorithms. The second part of the chapter focuses on the theory behind Genetic Algorithms and an account of their most important features. Additional topics on GAs are discussed in Appendix B.

Chapter 3 presents a preliminary optimization case. This study introduces the optimization of the parallel architectures design, being a problem of reliability and redundancy allocation plus test intervals. The multi-objective optimization problem is solved by the Fonseca & Fleming MOGA, which is described here. Dependability modelling is made using fault tree analysis with house events, and the first lifecycle cost model is formulated. This is used throughout the entire thesis with relevant adaptations in subsequent chapters.

Chapter 4 analyzes the system design optimization introducing diverse redundancy as a defence against Common Cause Failure. Since there are several different technological choices per subsystem, the fault trees become much more complex, and thus a cut set reduction algorithm is implemented for the dependability measures quantification. The β factor model is analyzed and applied together with some modifications. Finally, the application case compares the results of using diverse and non-diverse implementations.

Chapter 5 investigates the optimization of proof testing policies for Safety Instrumented Systems. A comprehensive review of the current standards and practice is given, which guides the optimization problem. In order to include its effects realistically a new model for time dependent probability of failure on demand has been developed, and it is presented here. The effects of testing strategies and intervals are analyzed, being the variables used during the optimization. The multi-objective algorithm NSGA-II is thoroughly introduced in this chapter. The results of this are presented towards the end of the chapter.

Chapter 6 studies the introduction of MooN voting redundant architectures in the optimization of SIS. Here the relevant parts of the entire methodology previously developed are integrated, including the PFD time dependent model of Chapter 5. A large part of the chapter is dedicated to the extension of the PFD(t) and STR models for inclusion of MooN voting architectures. Two application cases are studied, one of optimization of design and another one of testing policies. The benefits of introducing MooN voting subsystems in both optimization of system design and proof testing are addressed, looking for the achievement of better balanced solutions with respect to the three objectives.

Finally, Chapter 7 presents the concluding remarks. The chapter makes a summary of the thesis, giving a revision of the optimization approaches and their resolution. It also comments on the importance of multi-objective optimization of Safety Instrumented Systems. To conclude, what are considered to be the main achievements of the thesis are highlighted and the possible avenues for future work are outlined.

CHAPTER 1

Safety Instrumented Systems and RAMS+C modelling

Safety Instrumented Systems (SIS) are widely used in several industry sectors. The name is specific to the process industry, but similar systems are used for other sectors, such as the nuclear industry. The resilience of SIS against faults is provided through fault tolerant architectures. These subjects are reviewed in this chapter. The analysis of safety systems relies largely on reliability theory, on which dependability analysis is based. The most influential international standard for safety systems in general is IEC 61508, so its analysis is presented here. Following this, the modelling of dependability and lifecycle cost is reviewed, together with the main issues concerning SIS.

1.1. SAFETY INSTRUMENTED SYSTEMS

It is known that the failure of a safety critical control system to perform its intended function could result in loss of the assets of a company, widespread damage to the environment, harm to personnel and people around the facilities, and even loss of life. When safety is achieved by means of the correct operation of a system or equipment, e.g. a safety instrumented system, it is said that “*functional safety*” is being used.

Safety Instrumented Systems (SIS) pertain to the generic category established by the international standard IEC 61508 (IEC, 1998-2005) as safety-related systems. “Safety Instrumented System” is a term more currently used for process industry. It is defined by IEC 61511 (IEC, 2003), a standard derived from IEC 61508 specific for this sector, as an “*instrumented system used to implement one or more safety instrumented functions. A SIS is composed of any combination of sensors, logic solver and final elements*”. Notice that all instruments located in the plant grounds are designated with the generic name of field instruments; basically the sensors and final elements: transmitters, valves, etc.

A SIS has the objective of detecting and preventing plant hazardous conditions, which if they were not mitigated could develop into catastrophic events which could have consequences such as loss of assets and production, widespread damage to the environment and loss of life. Gruhn & Cheddie (1998) give another definition: Safety instrumented systems are those “*designed to respond to conditions of a plant that may be hazardous in themselves or if no action were taken could eventually give rise to a hazard. They must generate the correct outputs to prevent the*

hazard or mitigate the consequences”. In the process industry, SIS are usually implemented as Emergency Shutdown Systems (ESD), although they might be used for other applications, such as fire and gas detection (F&G). Each function that a SIS implements is called a “safety function”. They are normally implemented in low-demand mode of operation (i.e. they are in standby and operate only as a response of a demand, not continuously), with their architectures limited to a few practical options.

A process plant usually has several layers of protection. A layer of protection is a measure put in place as a defence to reduce the risk presented by the plant. It performs its function in a hierarchical fashion, with the aim of maintaining the safe condition of the plant after the previous protection layer has failed to do so. Examples of protection layers can be: the basic process control system, the ESD system, the pressure relief valves and the active and passive fire protection systems (Fig. 1.1). These protection layers usually would take action in the mentioned order.

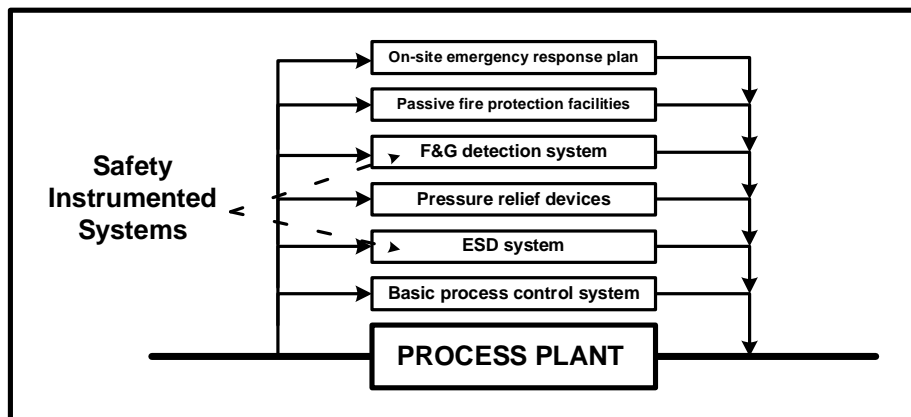


Figure 1.1. Position of SIS with the plant protection layers

An example of SIS used in this thesis is a protection system against high pressure and temperature of a chemical reactor (Fig. 1.2). The system is composed of four subsystems: Temperature measurement and transmitter (TT), pressure measurement and transmitter (PT), a safety controller or logic solver (LS) and a valve acting as the final control element (FC). Upon detection of either high temperature or pressure the safety system cuts the reactor supply off in order to prevent a runaway reaction. A pertinent clarification is that a transmitter usually includes the measurement sensor plus a transmission system for the measurement signal. This is the reason why in the safety function only transmitters are mentioned sometimes, omitting that

it includes the sensor in the same device. Transmitters have the function to send the measurement signal to the control room, where the controller is usually located.

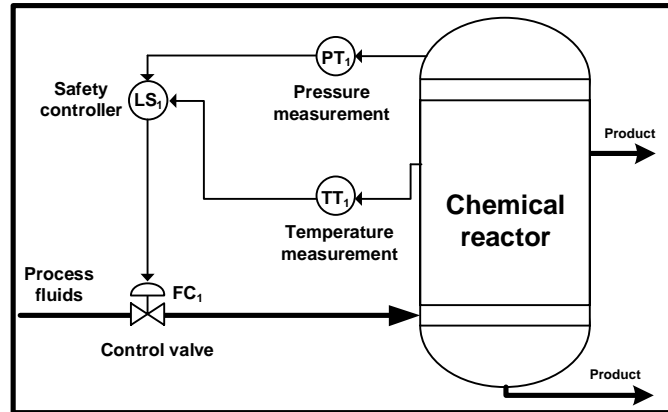


Figure 1.2. Example of SIS: Chemical reactor protection system.

The design of SIS entails achievement of some minimum levels of safety integrity, as required by IEC 61508. The safety integrity requirements include the restriction of the system probability of failure on demand (PFD_{avg}) to a maximum target limit and the compliance with some minimum levels of fault tolerance. Fault tolerance, the capacity of a system to prevent single faults escalating into system failures, is usually achieved by some form of redundancy. The most basic forms of redundancy are hardware and software redundancy. The implementation of hardware redundancy implies the use of extra equipment or parts, which the system would not normally need to perform its function, to tolerate potential faults.

Plain identical hardware redundancy may, however, introduce some collateral issues that cannot be overlooked, since they may lead to overoptimistic designs. Several identical redundant components are sensitive to stress factors that may lead to their simultaneous failure. This is known as Common Cause Failure (CCF). In consequence, CCF can be counteracted by implementing redundancy with components technologically diverse components; i.e. diversity in redundancy.

1.2. HARDWARE FAULT TOLERANCE

Safety related systems must achieve high levels of dependability. In order to ensure that dependability attributes (e.g. reliability, availability) meet the required specifications, some measures to prevent faults affecting the system must be put in place. This is addressed using two approaches: fault prevention (which encompasses fault avoidance and error removal) and fault

tolerance (Laprie, 1985). Fault prevention intends to avoid faults occurring or being introduced into the system as far as possible. Fault tolerance is a measure to prevent that faults that take place during service provoking a system failure. These two approaches are complementary. As Jalote (1994) stated, "*Fault prevention methods focus on methodologies for design, testing, and validation; whereas fault tolerant methods focus on how to use components in a manner that such failures can be masked*".

Fault tolerance is always achieved by means of some form of redundancy. The implementation of redundancy requires usage of extra equipment or parts that the system would not normally need to perform its function, but are used to tolerate potential faults. The most common form of redundancy is hardware redundancy, which makes use of extra hardware or replicate components.

A system is considered to be fault tolerant if it can prevent the presence of faults in the system by using redundancy. This prevents a component failure from producing a failure at higher levels. The system must have the capacity to avoid a failure of one component or module affecting the external behaviour of the entire system.

Whilst identical redundancy aims to address the problem of random failures, the technique of diversity is usable for tackling the problem of common cause failures as well. In systems with diversity in place a function is implemented in two or more different ways, in the hope that the same fault is not present in two diverse technologies or techniques. Diversity of design can be utilised for both hardware and software redundancy. When using redundancy and diversity together it is possible to address both random and (some) design faults. However, mistakes within the specification are not covered, and they must be addressed by other means of fault management.

Storey (1996) states that there are three different forms for implementing hardware redundancy: static (or passive), dynamic (or active) and hybrid. Static redundancy is based on fault masking, which means that the system "masks" the faults without taken any specific action, preventing the faults from resulting in further errors. Dynamic redundancy relies on fault detection followed by some action to counteract its effects removing the faulty hardware from the system. Hybrid redundancy relies on a combination of the two.

Static redundancy is usually implemented in two forms: using triple redundancy or using more than three redundant modules (N-modular). Figure 1.3 shows these two architectures. Static redundancy can also be implemented with double redundancy. However the effectiveness of

fault masking depends on the failure mode to be tolerated and the voting configuration. This is because if one channel fails it is not straightforward for the voter to determine which channel has failed and which has not when there is a discrepancy between them. This is further analyzed in Chapter 6, section 6.2 for SIS.

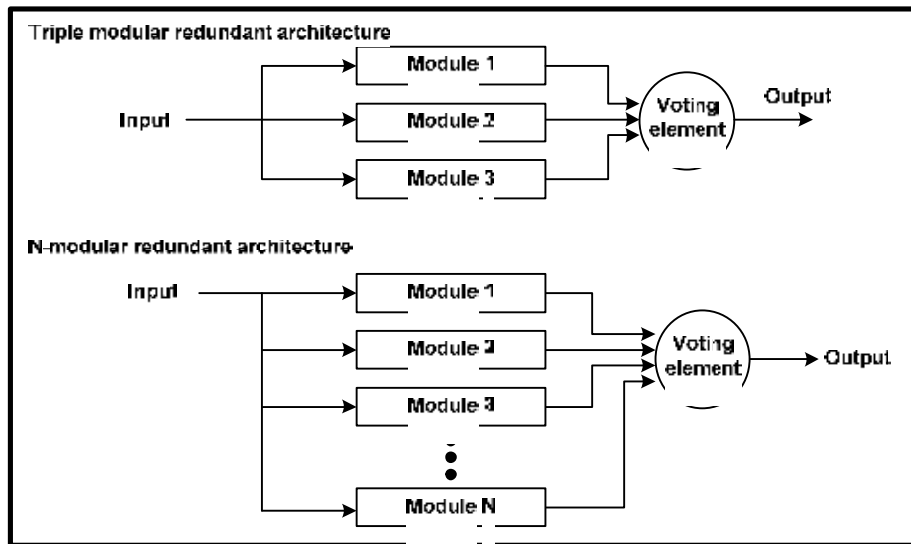


Figure 1.3. Examples of static fault tolerance

According to Storey, dynamic redundancy is based on fault detection, location and recovery to provide fault tolerance. Because this technique does not seek to mask faults, it does not prevent temporary errors occurring. Therefore, it is used where the error can be tolerated until the system reconfigures itself and eliminate the fault in a reasonable amount of time. The simplest way is to provide two redundant modules, one of them in standby. Basically, this redundancy scheme can be implemented using two modules being supervised by a fault detection mechanism, and a switching element that can select both of their outputs.

Fault tolerance is a term used mainly for computer systems. Nowadays many of the logic solvers used for SIS are computers (e.g. Programmable Logic Controllers - PLCs). However, fault tolerance is also implemented when redundancy is added to the field instruments (e.g. measurement devices, valves, etc). To simply connect two transmitters in parallel provides some degree of fault tolerance. Simple parallel redundancy is not usually mentioned in literature about fault tolerance. However, a parallel system actually makes a fault tolerant architecture if one or more parallel channels are there to only back up another in case of failure.

1.3. ARCHITECTURES FOR SIS

As stated before, a SIS is basically a combination of field instruments (sensors and final control elements) and a logic solver. As it will be detailed below, hardware redundancy in process industry SIS is practically limited to parallel and majority voting schemes with a small number of components (usually up to four).

1.3.1. Voting architectures

Voting architectures for SIS are usually limited to structures with some few components. For example, regarding field instruments, Gruhn & Cheddie (1998, 2005) identified the redundancies 1oo1D, 1oo2D and 2oo3 for sensors, while for final control elements they mentioned the schemes 1oo1, 1oo2 and 2oo2. Goble & Cheddie (2005) presented practical examples for the same sensor architectures. Leinum (1992) studied the redundancy of heat (fire) detector systems for an Emergency Shutdown System, and compare the metrics of several sensor voting schemes, such as 1oo2, 2oo2, 1oo3, 2oo3, 1oo4 and other kinds of combinations. Bodsberg & Hokstad (1995) showed sensor redundancies with up to 8 units, although it is not clear if the example was taken from real practice. CCPS¹ (2007) identifies the following voting architectures as the most common: Sensors: 1oo1, 1oo2, 2oo2 and 2oo3; Logic solver: 1oo1, 1oo2, 2oo2 and 2oo3; Final elements: 1oo1, 1oo2 and 2oo2. However, it recognizes that in some cases 2ooN with large N are used, usually “*where the unacceptable process condition can occur in multiple distinct locations*”.

The case of logic solver architectures is more complex, and it has received much more attention. Fault tolerant programmable controllers for safety applications in the process industry were examined firstly by Frederickson (1990) and Frederickson & Beckman (1991). They made a comparison of dual and triple PLC systems with different Input/Output redundancies (simple, dual and triple). Gruhn (1996) made a comparison of three technologies of logic solver: relay, PLC and Triple Modular Redundant (TMR). Goble et al. (1998) introduced the analysis of the diagnostic coverage for Programmable Electronic Systems (PES), which added the term “D” to the voting architectures. They analyzed the case of 1oo1D and 1oo2D architectures, where the “D” indicates that the diagnostic circuitry can be used to modify the voting output of the system in order to convert dangerous failures into safe ones. Goble (1998) developed dependability models for several different common architectures for PES: 1oo1, 1oo2, 2oo2, 1oo1D, 2oo3, 2oo2D, 1oo2D. These models were later incorporated into practical process industry examples (Goble & Cheddie, 2005). The standard IEC 61508 Part 6 makes an analysis to obtain simplified equations (by Reliability Block Diagrams) for the following architectures: 1oo1,

¹ Center for Chemical Process Safety

1oo2, 2oo2, 1oo2D and 2oo3.

IEC 61508 (and IEC 61511) refers to these architectures with the generic name of MooN (M-out-of-N) systems, which is widely accepted in the process industry. IEC 61511 defines MooN as a “*Safety instrumented system, or part of thereof, made of “N” independent channels, which are so connected that “M” channels are sufficient to perform the safety instrumented function*”. This term is also used by the PDS² method (Hauge et al., 2006a). According to CCPS (2007) “*N” designates the total number of devices (or channels) implemented; “M” designates the minimum number of devices (or channels) out of N required to initiate, take, or maintain the safe state*”. What the IEC standards do not explicitly mention (but it is clear from the examples given by Gruhn & Cheddie (1998, 2005), Goble (1998) and Goble & Cheddie (2005)) is that when the sensors and logic solver redundancy is more than 1, the MooN architecture is subject to majority voting, where M out of the N units need to vote for the protective function to be commanded. It can be said that a simple parallel redundancy is a case of MooN where M=1. Therefore, it can be concluded that SIS are usually implemented using simple parallel and MooN majority voting architectures.

Goble et al. (1998) show that in voting architectures with diagnostics (MooND) the diagnostic circuit can disconnect the faulty unit in order to change a dangerous failure into a safe failure. This idea was incorporated by Goble in his analysis of architectures. It is clearly seen that the diagnostic circuitry influences the voting outcome by its action. Therefore, it can be said that the term MooND implies that the diagnostic circuit can take part in the voting outcome (in substitution of the faulty unit). This would not apply to units that include automatic built-in diagnostics that only announce a fault. Notice that Gruhn & Cheddie (1996) and Goble & Cheddie (2005) present examples of MooND architectures for smart sensors as well (sensors with embedded microprocessors for added functionality and diagnostics).

It can be concluded that for sensor and logic solvers the most used architectures are parallel and voting MooN and MooND (this last only for logic solvers and smart sensors). For final control elements not many architectural examples have been found, the most common being simple parallel.

² Acronym for *Reliability of computer based systems* in Norwegian

1.3.2. Technologies used for SIS

The logic solvers for SIS have evolved from the original use of simple relays and switches, to solid-state devices, then integrated circuits with programmable capability, and finally dependable PLCs (Adamsky, 1991). They all have advantages and disadvantages, and although PLCs are highly used, the other options are still valid. Gruhn & Cheddie (1998) offer an account of them. Goble & Cheddie (2005) and CCPS (2007) in addition describe and analyse several devices for field instruments.

A PLC is a computer-based system, where a processor executing the safety functions is integrated with Input/Output (I/O) modules to provide control capability, and sometimes communications systems for interfacing to other systems. PLCs provide the advantages of programming capability, flexibility and interfacing to other systems (e.g. the control systems). Not all the PLCs used in industry for protective purposes are fail-safe and fault tolerant, but those utilised for highly critical applications are actually required to be. There are some other PLCs specially developed for implementing protective systems, called safety PLCs. They have a much higher diagnostic coverage and redundancy to increase fault tolerance. Three of the most widely used fault tolerant architectures are: Triple Modular Redundant (TMR), Duplex with extensive diagnostics and Quad Redundant. A study made by ARC³ (1999) presents a brief description of the described PLCs as well as which are the main suppliers in the market.

Technologies for sensors can be very varied according to the required measurement. Four of the most common variables in process industry are flow, level, temperature and pressure. The last two are some of the most critical in hazardous processes. Dependant on the device a sensor is attached to, in order to send a signal to the controller, there are several options:

- Electromechanical switches: The sensor is attached to a contacts switch that changes its position (open or closes) when some limit is reached in the measured variable.
- Conventional transmitters. The sensor is connected to an electronic or pneumatic device that amplifies and transmits an analogue signal representing the measured variable. They are used for transmitting the signal over longer distances, which is very common when the control room is far from the measuring point.
- Smart transmitters. They are electronic transmitters with an embedded microprocessor for enhanced functionality, which includes better diagnostic coverage. They usually send a coded analogue signal.

Final control elements, those which execute the commands of the controller, are very varied as

³ ARC Advisory Group

well. In the process industry the most common are valves and electric motors, and alarm devices (e.g. beacons, lights) for safety systems. Valves for safety systems are usually valves with only two positions (open and closed), called “on-off” valves. Their reliability is different depending on the kind of valve and the actuator used to command it.

1.3.3. Energized and de-energized systems.

The majority of safety instrumented systems are implemented as normally energized and de-energized to actuate (to trip) (Gruhn & Cheddie, 1998). These are called de-energized-to-trip systems. This is usually safer because if the power supply to the device fails, the system goes to a fail-safe position (i.e. self-revealed failure). An example is given by CCPS (2007) of a high level switch, where the switch has closed contacts during normal operation. If the level exceeds the permitted level (set point), the switch would open, de-energizing the circuit, to indicate this condition and command the required response (e.g. closure of an emergency valve). Some other applications are opposite; i.e. energized-to-trip, (usually machine control). Since energized-to-trip systems are normally de-energized, and a failure on this state may not be self-revealed, they usually require additional hardware to monitor its state; for example to monitor the continuity in the circuit.

1.4. RELIABILITY AND SAFETY CONCEPTS

In a broad sense, reliability deals with the ability of an item or system to perform its intended function. Reliability has become a term that encompasses several meanings, and because of this it has become somewhat confusing. Reliability engineering is a branch of engineering, a part of probability and statistics and also an attribute or measure.

At the lowest level, reliability is defined as the probability of a component or system to perform its intended function during a specific period of time and under a given set of conditions. It is therefore a measure of trustworthiness.

The study of reliability has grown to include many different aspects: analysis, modelling, etc. It has produced a body of theory derived mainly from probability and statistics: Reliability Theory. The study and application of Reliability Theory to engineering structures or systems has become known as Reliability Engineering. According to Rausand & Hoyland (2004), structural reliability deals with the analysis of structural elements, like beams and bridges. Another different approach is system reliability, which is focussed on the reliability of systems composed of several components, based on the probability distribution function of the time to failure of these components. This is the approach adopted in this work.

System reliability is associated with several metrics, one of them is actually reliability, and this can be another source of confusion. Availability and maintainability are other metrics studied in reliability theory and engineering. This makes reliability a global concept. It is for this reason that the author prefers to use the term Dependability to make reference to attributes such as availability, reliability and safety, as formulated by Laprie (1992). Rausand & Hoyland (2004) designate dependability as a “*collective term to describe the availability performance and its influencing factors*”. According to Avizienis et al. (2000), “*dependability is the ability to deliver a service that can justifiably be trusted*”. Dependability has several attributes: availability, reliability, safety, security, maintainability, etc. These attributes can become competing objectives, as remarked by Despotou & Kelly (2007), which results in “*inevitable trade-offs*”.

Another collective term is RAMS, which stands for reliability, availability, maintainability and safety. Martorell et al. (2005) added the important cost factor when analyzing systems, becoming RAMS+C. It should be noted that RAMS is therefore a subset of dependability.

Safety is the freedom of conditions that can cause accidents or losses (Rausand & Hoyland, 2004; Leveson, 1995). CCPS (2007) provides a less absolute definition: “*The expectation that a system does not, under defined conditions, lead to harm people, either directly or indirectly*”. The attribute of safety enforced by a system is designated as system safety.

Safety and reliability engineering share techniques and theory, but they are not the same. Since reliability theory is an older discipline, many of its foundations are used in safety engineering.

Leveson (1995) argues that it is often assumed that reliability and safety are synonymous, but that this is true only in special cases. Certainly, safety and reliability overlap, but they are not the same. Consider that accidents can occur without any component failure, and components may fail without resulting in accident. In addition, increasing reliability can increase safety, but in some other cases it can actually lead to situations that may reduce safety. This is quite true in safety systems, where reliability and safety can even be in conflict, depending on the measure used for reliability, as we will see later in Chapters 3 to 6. Reliability engineering aims to reduce failures, but this has a positive impact on safety only when this reduction affects failures that lead to hazards. This leads us to think that there are some failures that can lead to accidents, and there are others that will not. It would be therefore necessary to use two different metrics to measure the effects of those two different types of failures. This is actually the approach used in analyses of Safety Instrumented Systems.

1.4.1. Reliability

Reliability is the probability that an item will perform its intended function for a specific period of time under specific conditions. It is thus a probability of non-failure, or survival. Unreliability is the opposite, being the probability of failure for a specific time under specific conditions.

$$R(t) = 1 - \int_0^t f(t)dt \quad (1.1)$$

$$F(t) = 1 - R(t) \quad (1.2)$$

1.4.2. Failure Rate

This is defined as the number of failures per unit time of a sample of identical items. The Bathtub curve shown in Figure 1.4 represents an idealization of the typical behaviour of the failure rate of many devices. It is split into three periods: The burn-in period (infant mortality), where the failure rate is high and decreasing since many weak items with manufacturing or other faults fail. The useful life time section has as characteristic a practically constant failure rate. It is during this period of time that the components fail randomly, caused by external loads. The last part of the curve corresponds to the wear-out zone, where the failure rate increases due to the items ageing and their useful life ends.

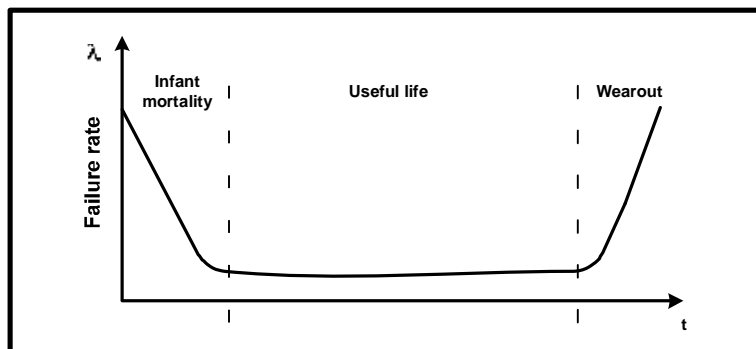


Figure 1.4. Bathtub curve model of failure rate

The constant failure rate is a valid assumption for many devices, especially electronic ones. Some other types of devices, such as mechanical equipment, present a typical decreasing failure rate. However, the constant failure rate would be a conservative worst-case assumption, and can still be used (Goble, 1998). The failure rate equals:

$$\lambda(t) = \frac{f(t)}{R(t)} \quad (1.3)$$

The constant failure rate leads to an exponential distribution (Lewis, 1996), which gives a cumulative distribution function:

$$F(t) = 1 - e^{-\lambda t} \quad (1.4)$$

$$R(t) = e^{-\lambda t} \quad (1.5)$$

Assuming that λt is small ($\ll 0.1$), the probability of failure $F(t)$ can be approximated by the rare event approximation as:

$$F(t) \approx \lambda t \quad (1.6)$$

1.4.3. Availability

This is the probability of an item being capable of (available for) performing its intended function at a given time. This is an instantaneous measure, and it can be averaged over a specific period of time T to give average availability. In contrast, unavailability is the probability that the system is not capable to perform its intended function at a given time. This is a measure of failure.

$$Q(t) = 1 - A(t) \quad (1.7)$$

$$A_{avg} = \frac{1}{T} \int_0^T A(t) dt \quad (1.8)$$

Reliability is a measure mostly used for non-repairable systems, with availability being used for repairable systems. Notice that an item can be unavailable at a specific moment either because it has failed (unreliability) or because it is under maintenance (preventive maintenance, test or repair). This means that availability is a function of both the internal reliability of the item and maintainability. In summary, unavailability is a measure of downtime.

1.4.4. System reliability

The total reliability of a system composed of several components can be quantified considering the particular structure they form. The most basic structures are series and parallel. Another very used structure is the k -out-of- n . In this section we use Reliability Block Diagrams (RBD) for system reliability quantification. It is a method that illustrates conveniently simple structures. Other methods may be more suitable for more complex structures. These are discussed in Section 1.6.

Reliability Block Diagrams represent the logical relationship between the components for successful functioning of the system. Each square block represents one component. Figure 1.5 shows the RBDs for examples of the three basic structures with three components.

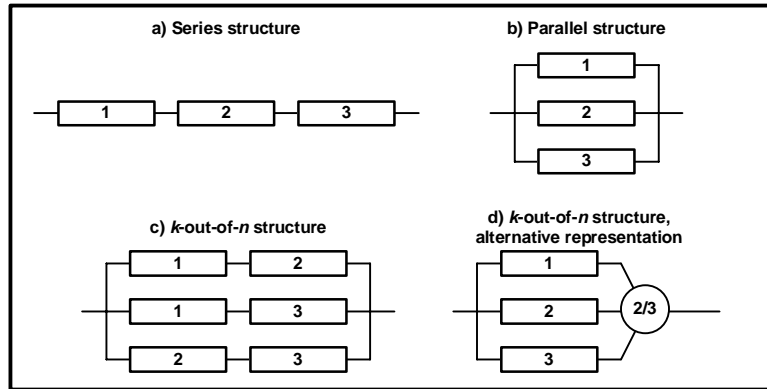


Figure 1.5. Basic system structures

- **Parallel structure.** A parallel structure (Fig. 1.5b) corresponds to a system that functions if at least one of its components is functioning. Eq. (1.10) expresses the reliability of a parallel structure.

$$R_{\text{sys}} = 1 - (1 - R_1) \cdot (1 - R_2) \cdot \dots \cdot (1 - R_n) = 1 - \prod_{i=1}^n (1 - R_i) \quad (1.10)$$

For two components this can be reduced to:

$$R_{\text{sys}} = R_1 + R_2 - (R_1 + R_2) \quad (1.11)$$

Notice that in Eq. (1.10) the term $(1 - R_i)$ is equal to the component unreliability, which can be directly substituted in the equation if the value is known.

- **k-out-of-n structure.** The k -out-of- n is a structure widely used in SIS. A k -out-of- n : G corresponds to a system that functions if at least k out of the total n components work. The letter G indicates a “good” condition. In contrast a k -out-of- n : F system is that that fails if at least k out of the n components fail. Notice that when no letter G or F is indicated it is assumed that it is a k -out-of- n : G structure. A Moon voting system is equivalent to a k -out-of- n : G structure plus a perfect voter. This type of system will be studied in detail in Chapter 6. Figure 1.5c shows the example of a 2-out-of-3 structure, while Figure 1.5d shows an alternative representation for the same example.

The reliability of a k -out-of- n : G system in its most basic form (when all the n components are identical) can be quantified based on the binomial distribution:

$$R_{\text{sys}} = \sum_{i=k}^n \left(\frac{n!}{i!(n-i)!} \right) R^i (1-R)^{n-i} = \sum_{i=k}^n \binom{n}{i} R^i (1-R)^{n-i} \quad (1.12)$$

Since reliability and availability are both probabilistic measures of success, Eqs. (1.9-1.12) are equally applicable to system availability quantification. System unavailability can also be worked out from these equations. For a more detailed treatment the interested reader is referred to Rausand & Hoyland (2004). There are several more complex structures that for simplicity are omitted here. A detailed analysis can be found in Kuo & Zuo (2003).

1.4.5. Diagnostic Coverage

Modern safety systems usually have an in-built hardware and/or software mechanism for automatic detection of internal failures. This is called “diagnostics” (CCPS, 2007). The diagnostic coverage (ε) is the fraction, usually expressed in percentage, of the total failure rate that this diagnostic mechanism can actually detect. It therefore splits the total failure rate between detected and undetected failures.

$$\lambda^{Detected} = \varepsilon \cdot \lambda^{Total} \quad (1.13)$$

$$\lambda^{Undetected} = (1 - \varepsilon) \cdot \lambda^{Total} \quad (1.14)$$

1.4.6. Common Cause Failure

This is the failure of more than one item due to the same stress or cause (Goble, 1998). Common Cause Failure (CCF) is a phenomenon that negates the benefits of redundancy, and it is therefore an important problem to address. This phenomenon has been given great attention in this work, and it will be considered in detail in the following chapters. There are several models to quantify CCF. The most popular is the β factor model (Mosleh et al., 1988). The β factor represents the fraction of the total failure rate that can be attributed to a common cause, affecting an entire group of items. This factor splits the total failure rate into common cause failures and independent (or normal) failures:

$$\lambda^{CCF} = \beta \cdot \lambda^{Total} \quad (1.15)$$

$$\lambda^{Normal} = (1 - \beta) \cdot \lambda^{Total} \quad (1.16)$$

1.4.7. Failure classification

In the most general sense, there are two categories of failures: Random (or physical) failures and systematic (or functional) failures. Random hardware failures are caused by the component ageing degradation (Hauge et al., 2006a), which can be accelerated by stress factors (CCPS, 2007). Systematic failures are caused by practically any other cause than degradation. Hauge et al. (2006a) classify them as stress failures, design failures and interaction (operational) failures. Systematic failures can cause Common Cause Failures. They can also prevent a component from performing its intended function even when it is still able to operate. IEC 61508 recommends considering only random failures in the calculations given that systematic failures

cannot usually be quantified. However, as Hauge et al. (2006a) points out, IEC 61508 implicitly includes the quantification of some (undetermined) systematic failures on the method for quantifying hardware common cause failures. Figure 1.6 illustrates the failure classification used in this thesis.

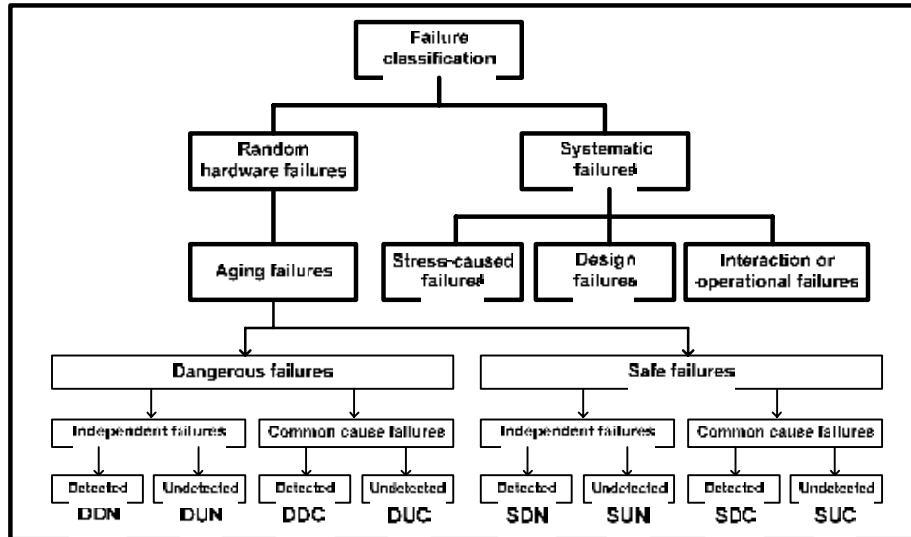


Figure 1.6. Failure classification and nomenclature

1.4.8. Random hardware failure modes

As previously mentioned, not all hardware failures have the same effects. In safety systems, some failures can lead to hazardous situations, and some others can cause false activations (but without causing any hazard). Thus, there are two main types of failure modes: dangerous and safe. Dangerous failures have potentially hazardous consequences, i.e. failure to perform the protective function when required; while safe failures result in spurious activations of the system (spurious trips). Consider for example the protective systems given in Figure 1.2. A hazardous condition could be a very high reactor pressure. The sensor/transmitter could be faulty, and fail to detect that the pressure value has surpassed the set point. This is a dangerous failure, which could fail to stop a runaway reaction for example, with disastrous consequences. In contrast, it could be the case that there is actually no hazardous condition present in the reactor, no high pressure or temperature, and the sensor/transmitter falsely detects a high value of one of those variables. This would lead to the emergency isolation valve to be shut: a spurious reactor trip. This is a safe failure since no hazardous condition has been created or overlooked. It certainly causes production losses, or the entire upset of the plant, but it is still in a safe condition.

Consequently, the total failure rate must be split into safe (λ^S) and dangerous (λ^D) failure rates.

$$\lambda^T = \lambda^D + \lambda^S \quad (1.17)$$

Additionally, the system built-in diagnostic coverage (ε) divides both dangerous and safe fractions into those revealed by its action and those not (Eq. (1.18)).

$$\lambda^T = \varepsilon^D \cdot \lambda^D + (1 - \varepsilon^D) \cdot \lambda^D + \varepsilon^S \cdot \lambda^S + (1 - \varepsilon^S) \cdot \lambda^S \quad (1.18)$$

Observe that there is a different diagnostic coverage for dangerous and safe failures. This further splits the total failure rate into detected (λ^{DD} and λ^{SD}) and undetected (λ^{DU} and λ^{SU}) failure modes (Eq. (1.19)).

$$\lambda^T = \lambda^{DD} + \lambda^{DU} + \lambda^{SD} + \lambda^{SU} \quad (1.19)$$

Applying the fractions determined by the β factor (Eqs. (1.15-1.16)) to total failure rate Eq. (1.19), we finally include the CCF quantification:

$$\lambda^T = \lambda^{DDC} + \lambda^{DDN} + \lambda^{DUC} + \lambda^{DUN} + \lambda^{SDC} + \lambda^{SDN} + \lambda^{SUC} + \lambda^{SUN} \quad (1.20)$$

Where the failure rates are:

λ^T = Total failure rate

λ^{DDC} = Dangerous detected common

λ^{DDN} = Dangerous detected normal (independent)

λ^{DUC} = Dangerous undetected common

λ^{DUN} = Dangerous undetected normal (independent)

λ^{SDC} = Safe detected common

λ^{SDN} = Safe detected normal (independent)

λ^{SUC} = Safe undetected common

λ^{SUN} = Safe undetected normal (independent)

1.4.9. Risk reduction

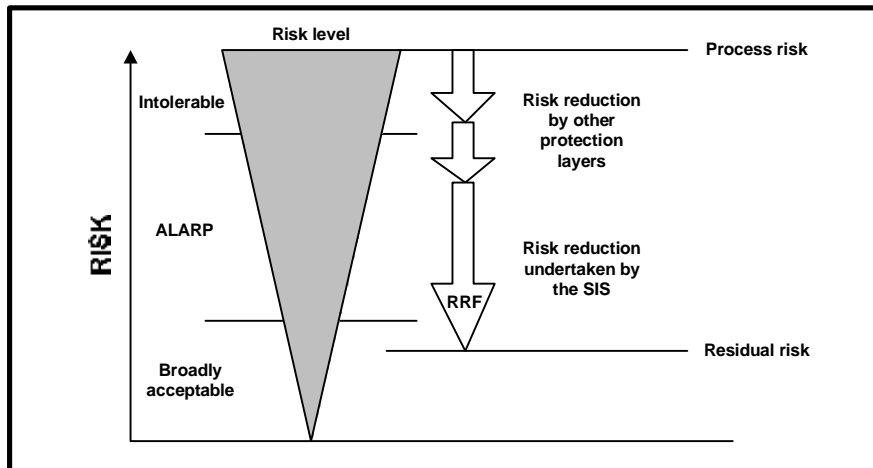


Figure 1.7. Risk reduction concept

The objective of a safety system is to reduce the risk presented by the plant to acceptable levels. The UK Health and Safety Executive established a suggested framework for risk criteria to be used in the UK (HSE⁴, 2001), called “the Framework for the Tolerability of Risk”. Figure 1.7 shows that the risk comprises three regions regarding the actual level of risk:

- Intolerable region. If the risk falls within this region, its level is intolerable, and further measures to reduce it must be implemented irrespective of their cost.
- ALARP region. In this region, the level of presented risk might be acceptable, but only if can be demonstrated if that it is “As Low As Reasonably Practicable” (ALARP). This means that it must be demonstrated that all the reasonably practical measures have been put in place to reduce the risk to be as low as possible. If this principle is demonstrated, the risk is considered tolerable. Determination whether a risk has been reduced to an ALARP level depends on several factors. Some guidance on the subject can be found in HID⁵ (2003).
- Broadly acceptable region. In this region the level of risk is negligible, and hence broadly acceptable. Therefore, no additional measure needs to be considered.

The risk reduction required to the SIS is estimated based on the difference existing between the actual risk and the risk considered acceptable. The conceived acceptable risk represents the target that the SIS must contribute to achieve; i.e. the risk target. Therefore, the Risk Reduction Factor (RRF) is a figure that specifies the factor to be applied for reducing the risk from the actual level to the target level. Figure 1.7 illustrates the concept. It shows that the risk reduction is achieved by the application of several layers of protection, where the SIS is one of them. The figure of the RRF may be determined for the entire risk or only for that addressed by the SIS.

1.4.10. Probability of Failure on Demand

Probability of failure on demand (PFD) is a measure of unavailability of the safety system for performing its protective action when required to do so. It is therefore a measure of loss of safety (Hauge et al., 2006a). It is also used by IEC 61508 and thus the metric of system safety used in this work. It is important to clarify that PFD does not refer to the probability of the system having a failure caused by the demand to actuate itself (i.e. the demand is the initiating event), which is another contribution to system unavailability used in other disciplines (Vaurio, 1995a). PFD can be a confusing term, and some other authors have proposed to use alternative names, like “*probability of not functioning on demand*” (Duduit et al., 2008).

$$PFD(t) = 1 - e^{-\lambda^D t} \approx \lambda^D t \quad (1.21)$$

⁴ Health and Safety Executive (UK)

⁵ Hazardous Installations Directorate (UK)

Since $PFD(t)$ is a measure of loss of safety, it is affected only by the dangerous failure mode λ^D . The PFD is a time dependent measure, and it is averaged over a defined time interval, which is usually assumed to be the test interval TI (CCPS, 2007), to become PFD_{avg} :

$$PFD_{avg} = \frac{1}{TI} \int_0^{TI} PFD(t) dt \quad (1.22)$$

Figure 1.8 shows the $PFD(t)$ behaviour. According to Eq. (1.21), it follows an exponential increment until the component is tested. Under the ideal assumption of perfect testing, the $PFD(t)$ is reduced to zero each test event. Another important measure is the maximum instantaneous PFD (PFD_{max}).

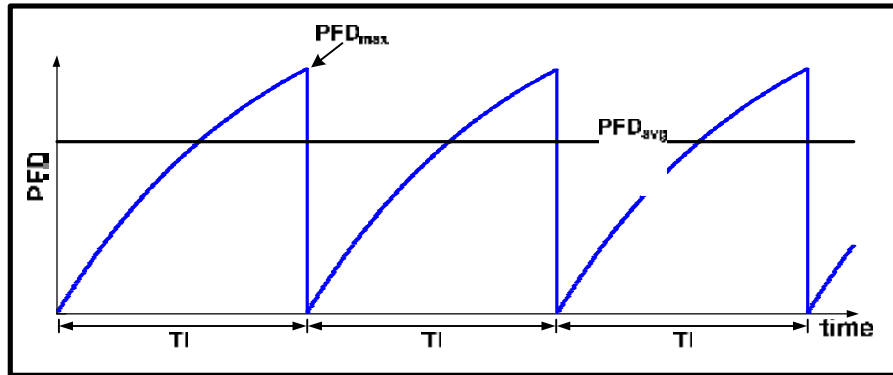


Figure 1.8. Time-dependent Probability of Failure on Demand

1.4.11. Spurious Trip Rate

Spurious Trip Rate (STR) is the frequency of spurious activations of the safety system per time unit (Hauge et al., 2006a). For one single system is equivalent to its safe failure rate.

$$STR = \lambda^S \quad (1.23)$$

There is no unanimous consensus about which metric to use for quantifying the effect of safe failures, since this is not a requirement of IEC 61508. However, IEC 61511 requires specification of a maximum STR for any system, and the PDS Method uses this as well as a safe failure metric (Hauge et al., 2006a). STR addresses the issue of reliability, and has an important impact on the trustworthiness that the user places upon the system and the overall system life cycle cost since it results in losses of production.

1.5. REQUIREMENTS OF THE STANDARD IEC 61508

The international standard IEC⁶ 61508 “Functional safety of electrical/electronic/programmable electronic safety-related systems” (IEC, 1998-2005) aims to provide a means of ensuring that safety is effectively reached based on functionality of electrical, electronic or programmable electronic systems. Since IEC 61508 is a generic document, non-specific to any industry sector, it is relevant to a large range of different sectors: aircraft industry, process industry, nuclear, automotive, marine, etc.

IEC 61508 comprises seven parts, being a fairly complex standard. However, the first three parts are the most important ones. It can be said that Part 1 addresses management, mainly non-technical, requirements, while Part 2 deals with technical requirements for the hardware realisation and Part 3 with technical requirements for software.

The basis of IEC 61508 is the establishment of the safety lifecycle and the definition of Safety Integrity Levels (SIL). The standard requires that every safety function must achieve a specific SIL, determined beforehand based on a previous risk assessment. The SIL is a quantitative index that indicates the acceptable probability of dangerous failure that a system can have to consider it appropriate for a given specific safety integrity requirement. Distinction is made between two different kinds of systems: low-demand mode and high-demand/continuous mode of operation. Safety Instrumented Systems are usually in low-demand mode of operation. This mode of operation is defined by IEC 61580-4 as the one “*where the frequency of demands for operation made on a safety-related system is not greater than one per year and no greater than twice the proof-test frequency*”. For these, the SIL levels are defined in terms of average probability of failure on demand (Table 1.1).

Table 1.1. SIL for Low Demand Mode of Operation (IEC 61508 Part 1)

SIL	PFD_{avg}
4	$\geq 10^{-5}$ to $< 10^{-4}$
3	$\geq 10^{-4}$ to $< 10^{-3}$
2	$\geq 10^{-3}$ to $< 10^{-2}$
1	$\geq 10^{-2}$ to $< 10^{-1}$

In addition to the requirement of bounded PFD_{avg} , the highest SIL that can be claimed for a subsystem’s combination for hardware is limited by architectural constraints, which are detailed in IEC 61508 Part 2. These constraints are the hardware Fault Tolerance (FT) and the Safe Failure Fraction (SFF). Table 1.2 shows the multiple combinations that the standard acknowledges. Notice that subsystems are treated as belonging to two different types: A and B.

⁶ International Electrotechnical Commission

Subsystems Type A must be composed of components that meet the following conditions: *“their failure modes must be well defined; their behaviour under fault conditions must be completely determined; and sufficient dependable failure data from field (must) show that the claimed rates of failure for (all) dangerous failures are met”* (IEC 61508 Part 2). Practically, a subsystem is considered to be Type B if it does not meet any of these conditions. Goble & Cheddie (2005) specify that examples of Type A devices are relays, solenoids, valves and simple electronic modules (e.g. conventional transmitters). Any device with embedded microprocessors (such as smart transmitters) or complex application specific integrated circuits is considered Type B.

Table 1.2. SIL architectural constraints (IEC 61508 Part 2)

Subsystem TYPE	A			B		
	0	1	2	0	1	2
Hardware FT						
SFF:						
< 60 %	SIL 1	SIL 2	SIL 3	N/A*	SIL 1	SIL 2
60% -<90%	SIL 2	SIL 3	SIL 4	SIL 1	SIL 2	SIL 3
90% -<99%	SIL 3	SIL 4	SIL 4	SIL 2	SIL 3	SIL 4
≥ 99 %	SIL 3	SIL 4	SIL 4	SIL 3	SIL 4	SIL 4

*Not allowed

The Fault Tolerance (FT) is the number of faults that a subsystem can tolerate before resulting in system failure. This means that with a Fault Tolerance of N, N+1 faults would cause the system to fail. Notice that the maximum Fault Tolerance level contemplated in Table 1.2 is two. This may be considered as a restriction to claim higher SILs for higher levels of Fault Tolerance. The effects of this restriction will be seen in the discussion of results in Chapter 3. Safe Failure Fraction (SFF) is the percentage of subsystem failures considered safe: i.e. that they are either safe mode or they are revealed by the diagnostic coverage (Eq. (1.24)).

$$SFF = \frac{(\lambda^T - \lambda^{DU})}{\lambda^T} \quad (1.24)$$

If a safety function is implemented using only one single channel the highest SIL level to be claimed is that of the subsystem with the lowest SIL. On the other hand, if a safety function were implemented using multiple channels of subsystems, the SIL would be determined making an analysis of the combination. For example, consider a subsystem of two smart transmitters in parallel, each having a SFF of 92%. Since the overall arrangement would tolerate the failure of one transmitter (still being able to perform its safety function), the Fault Tolerance is 1. Smart transmitters have a microprocessor embedded, thus classified as Type B devices. For example, Table 1.2 indicates that for a subsystem with Type B devices, Fault Tolerance of 1 and SFF of 92% a maximum SIL of 3 can be claimed.

It is important to notice that SFF is a function of system architecture. Therefore, it is determined based on the specific combination of components. When the redundant channels or components of a system have different SFF (i.e. different failure rates), the total SFF will be a result of their combination. Thus, the failure rates used in Eq. (1.24) shall be at system level.

Goble & Cheddie (2005) substituted the different failure modes rates (Eq. (1.19)) into Eq. (1.24), and demonstrated that:

$$SFF = \%Safe + (1 - \%Safe) \cdot \varepsilon^D \quad (1.25)$$

Where $\%Safe$ is the percentage of only safe mode failures:

$$\%Safe = \frac{\lambda^S}{\lambda^T} \quad (1.26)$$

This clearly shows that the SFF depends on the ratio of the safe failure rate to the total failure rate. Note that the SFF is basically the percentage of the total failure rate that does not cause a dangerous undetected failure. Therefore, two components with very different total failure rates (one with very high rate and one with very low total failure rate) can have the same SFF.

Considering this, it is possible to say that the architectural constraints imposed by the IEC 61508 tables aim to ensure that an over-optimistic failure rate data is not used. This prevents the claim of unrealistically high SIL levels for designs without an adequate level of redundancy. This is permitted by the fact the FT is independent of the total failure rate, and the SFF is a metric of the ratio of failure rates rather than the total.

The use of PFD_{avg} as the standard metric has been questioned by Signoret et al. (2007) and Dudit et al. (2008), because the $PFD(t)$ can reach a maximum value significantly higher than the PFD_{avg} , and spend a large share of the time in a SIL region one value higher than the one determined by the PFD_{avg} . However, there is not a generalized consensus about the use of any other metric, and the PFD_{avg} continues to be accepted.

There also is some controversy about the real usefulness of the SFF and its precision to fulfil its intended effect (Signoret, 2007; Lundteigen & Rausand, 2008; Yoshimura & Sato, 2008). Their arguments are well founded, and it is probable that during the revision of the standard some modifications are introduced (due in 2010). However, so far the standard is still valid as it is, and therefore we take into account its requirements as they established at the moment.

In summary, determination of the final SIL, which implies calculation of PFD_{avg} , FT and SFF, requires a sufficient level of modelling detail (see IEC 61580-2 Clause 7.4.3.2.2). This means

that the following factors must be included:

- Specific architecture (redundancy scheme)
- Hardware type: Type A or B
- Multiple failure modes (safe and dangerous failure rates)
- Diagnostic coverage (share of detected and undetected failures)
- Common Cause Failure (β factor)
- Proof Test Intervals
- Repair times

1.6. DEPENDABILITY MODELLING

1.6.1. Modelling methods

Dependability can be modelled for safety analysis using various approaches. Analytical models are the basis for any quantification, and they are useful for analyzing time dependencies. However, they are only easily implementable for single components or for systems with few components (usually up to three). In addition, when increasing the modelling detail, including features such as CCF, diagnostics, and maintenance, analytic models become difficult to obtain and handle. Therefore, for more complex systems other probabilistic models are used. The most popular are combinatorial methods, mainly Reliability Block Diagrams (RBD) and Fault Tree analysis (FTA). These are static models that permit calculation of point or average values. RBD are generally useful for non-repairable systems. FTA can handle repairable systems, but for systems with sophisticated repair policies or time dependencies other more sophisticated methods are used. These are methods which can be termed “transitional”, such as Markov Analysis (MA), Petri Nets and Bayesian Networks. However, these methods have the problem of exponential growing complexity (i.e. 2^n potential states in Markov chains for n components, Signoret et al., 2007).

The next sections present a review of modelling techniques and their application in quantification of PFD_{avg} and STR. For the reader not familiar with these topics or looking for more detailed account, further details are given in Appendix A.

The main methods used for dependability measures in safety analysis are:

1. Simplified equations (SE).
2. Reliability Block Diagrams (RBD).
3. Fault Tree Analysis (FTA).

4. Markov Analysis (MA).
5. Petri Nets (Duduit et al., 1997, 2008).
6. Hybrid methods.

There are several comparative studies on dependability models. In general, these comparisons find that the diverse methods provide very similar results (ISA, 1999; Goble, 1998; Goble & Cheddie, 2005). Goble (1998) concluded that FTA and MA provide similar results for modelling Moon architectures; however the MA approach is advantageous for inclusion of time dependency and multiple failure modes interactions. Regarding RBD and FTA, Goble & Cheddie (2005) found that the main difference is that RBD focus on system success, while FTA focuses on system failure. They prefer FTA since it shows clearly the failure propagation mechanisms. Rouvroye & Brombacher (1999) showed that RBD gives very pessimistic results. FTA and some hybrid methods have approximately the same capabilities although FTA cannot include systematic failures. For these authors MA is the preferred method.

Andrews & Ericson (2000) compared FTA and MA for several design complexities. They concluded that FTA delivers either the same results or very good approximations to MA. Although MA is more exact, it is frequently necessary to exclude many contributing events in order to simplify the model, which converts it into an approximation. They point out that to create Markov models for systems that are not very simple is difficult and error prone, and for complex systems solutions can be only obtained by using numerical methods. FTA in contrast, is powerful for modelling large and complex systems easily, and their results are very good when small probabilities are involved (usual in safety systems). Bukowski (2005) also concluded that SE may lead to significant errors and that MA requires expert knowledge for their application.

IEC 61508 Part 6 suggests a PFD_{avg} quantification method based on simplified equations (obtained from RBD). It presents the disadvantages of SE mentioned above since it seems to be oversimplified and not adequate for detailed analysis. Hauge et al. (2006a) in the PDS Method presents a more refined technique based on calculation formulas. The method aims to include failure categories and causes excluded by previous techniques. It also presents a very novel approach to quantification of CCF by the β factor model, which is incorporated in this thesis.

In general, it seems that the only method that could be superior to FTA is MA, since this is capable of handling time-dependencies, apart from sequential failures. However, MA has the significant drawback of growing complexity, which increases exponentially with the number of components. Even modelling with more than two components can become unmanageable when

several failure modes are included. In addition, FTA has the advantage over MA that it is much easier and intuitive to construct and it provides a graphical representation of the failure mechanisms. Other methods such as Petri nets (Schneeweiss, 2001) can handle time dependencies, but they are also complex to construct and analyze. Sequential failures can be handled by dynamic fault trees (Dugan et al., 1992).

FTA is frequently preferred over RBD because it provides a more graphic understanding of the failure processes, and it focuses on failure rather than success probability, which is generally the approach from the safety view point (Andrews & Ericsson, 2000). Also, FTA is at a more researched and developed stage. In addition, Simplified Equations and the hybrid methods mentioned here (based on SE as well), present the disadvantage of oversimplification (IEC 61508-6) and inflexibility to accommodate changing conditions of design (Hauge et al., 2006a).

1.6.2. Overview of Modelling of PFD for SIL analysis

IEC 61508 established the use of the average Probability of Failure on Demand as the standard metric for loss of safety. The IEC 61508-6 method for quantification of PFD_{avg} is, as mentioned before, based on simplified equations derived from RBDs. It also provides tables for the PFD_{avg} of a few specific architectures for fixed combinations of failure rate, diagnostic coverage and β factor. IEC 61508 has been adopted as standard practice in many industrial sectors and different countries, since it provides an effective organizational framework for implementation and operation of safety systems. However, some authors have identified several limitations and inconsistencies, especially with the method for quantification of PFD and the evaluation of SIL levels (Zhang et al., 2003; Hokstad & Corneliusen, 2004; Signoret et al., 2007; Guo & Yang, 2007; Dudit et al., 2008; Lundteigen & Rausand, 2008a).

The main drawbacks of the IEC 61508 methodology are:

- It does not provide sufficient detail and explanation to implement it (Zhang et al., 2003; Guo & Yang, 2007). According to Signoret et al. (2007), it does not detail the method and underlying hypothesis to derive the given formulae, and therefore it is difficult to know under what conditions they are really valid.
- Zhang et al. (2003) found some inconsistencies in the calculation of the equivalent mean down time T_{CE} and T_{GE} terms, which is not clearly defined in the standard, and on which all the PFD_{avg} calculations are based.
- Insufficient failure taxonomy and definitions (Signoret et al., 2007; Hokstad & Corneliusen, 2004).
- Its capacity to handle test and maintenance procedures is very limited. Considering this, Signoret et al. (2007) proposes to extend the parameters included in the formulation.

- The use of architectural constraints as guidance for validation of SIL figures, specially the SFF (Signoret et al., 2007; Lundteigen & Rausand, 2008a).
- Limitations of the β factor model for evaluating accurately systems where the redundancy is larger than two, and when voting is introduced (Hokstad and Corneliusen, 2004).

The formulae provided by IEC 61508 are limited to a few architectures, and its tables limited to some specific combinations of component failure rates, diagnostic coverage and β factor. It would be necessary to derive new formulae for other architectures. Fundamentally, there is no flexibility for handling changing conditions, such as it would be needed in optimization studies.

Goble (1998) provides an alternative to modelling PFD_{avg} . As mentioned above, it uses either FTA or MA, providing the detailed modelling of several architectures, including safe and dangerous failures, CCF and diagnostic coverage. This is the base used in this thesis for fault tree modelling. Goble bases his modelling on using the failure modes taxonomy as presented in Eq. (1.20). He solves the FTs by deriving simplified equations and obtaining the average value by integration. The methods presented by ISA⁷ TR84.0.02 (ISA, 1999) (SE, FTA and MA) use the same failure modes taxonomy. The standard attempts to include systematic failures, but recognizes that it is very difficult to obtain statistical data for them. Some other authors (Knegtering & Brombacher, 1999; Rouvroye & Brombacher, 1999; Rouvroye & van den Blik, 2002) proposed other more complex Hybrid methods.

SINTEF⁸ proposed a new analytic method for quantification of reliability for process safety systems (Bodsberg & Hokstad, 1995, 1997). It is called the PDS⁹ Method. It explores the creation of an alternative failure taxonomy that relates directly failure cause, consequence and their means of improvement. This also addresses the need of using reliability calculations for LCC quantification in order to find cost-effective designs and operating philosophies. At a later stage (Hokstad & Corneliusen, 2004), they started to identify the weaknesses in the IEC 61508 methods, proposing their new failure classification and suggesting an extended and more precise β factor model for quantification of CCF of different MooN architectures (discussed in Chapter 4). The PDS method is more comprehensive than the one of IEC 61508-6. However, there exists the difficulty of getting explicit input data for all its parameters. The PDS method is classified as a hybrid method (it uses simple RBDs), but is largely based on SE, and therefore it presents some of their drawbacks (oversimplification, difficulty to handle test and repair, etc.). However, its contribution is enormous, especially considering its provision for quantification of CCF and

⁷ The Instrumentation, Systems and Automation Society

⁸ The Foundation for Scientific and Industrial Research (in Norwegian)

⁹ Norwegian acronym for "reliability and availability of computer-based safety systems"

independent failure rates of different MooN architectures. This has had large influence on the work presented in this thesis, from Chapter 4 on, and especially in Chapter 6. A more detailed description of the PDS Method is given in Appendix A, Section A.1.2.

Signoret et al. (2007) and Dudit et al. (2008) have proposed the use of fault trees for quantification of $PF_{D_{avg}}$ complemented with more powerful methods so that time-dependencies can be handled, and test and maintenance properly modelled. This constitutes hybrid methods based on fault trees. The idea is to introduce multi-phase Markov models or Petri Nets substituting sub-modules of the fault tree, included as basic events. In this thesis, a similar idea has been developed (Chapters 5 and 6), but introducing time-dependent analytical models that comprise entire subsystems.

1.6.3. Overview of modelling of STR for quantification of safe failures

The quantification of safe failures of SIS has traditionally not had as much attention as dangerous failures by the safety community. Since it is generally considered that safe failures do not have a direct impact on safety, it is usually overlooked in safety system analysis. However, this approach is starting to change (Lundteigen & Rausand, 2008b). The standard IEC 61508 does not include safe failure's consequence quantification, and therefore does not provide any quantification method. However, IEC 61511 (IEC, 2003) does include a requirement to specify a maximum level of STR for safety systems in the process industry. As it will be seen below, several related measures have been proposed as metric of safe failure consequences. STR has been adopted herein because it is the one that has become more widely accepted (e.g. IEC, 2003; Hauge et al., 2006a; CCPS, 2007; Lundteigen & Rausand, 2008b).

Goble (1988) includes quantification of Probability of Failing Safely (actually unavailability by safe failure) in the modelling with FTA and MA. The contribution of Goble is significant because he analyzes safe failures with the same level of detail as dangerous ones. The measure for safe failure included by ISA TR84.0.02 (ISA, 1999) is the Mean Time to Spurious Trip ($MTTF^{spurious}=1/STR$). Notice that ISA TR84.0.02 drops one level of modelling detail (i.e. separation by diagnostic coverage) in the Markov examples to avoid the complexity problem.

The PDS Method acknowledges the fundamental importance that the quantification of consequences of safe failures has for SIS, conceptualizing the STR as a measure of the system's ability to maintain safe production. Therefore, STR is a measure of loss of production. The method also establishes firmly the necessity of quantifying STR in order to keep a balance between loss of safety and loss of production, and not only as a secondary performance metric. It gives a set of generic equations for evaluation of STR for different architectures is given. The

method is oriented to contemplate the different effects of the specific MooN voting architecture, embedded in the modified β factor model.

Lu & Lewis (2006, 2008) presented analytical models for probability of spurious operation (and unavailability) based on the binomial distribution. Although the level of modelling detail is too basic, their contribution of this work is important because it introduces the evaluation of the relative time the system spends in normal operation and test and maintenance, during which its probability of spurious operation changes. This concept is considered in Chapter 6.

The first monothematic study about spurious activation of SIS was presented by Lundteigen & Rausand (2008b). This article defines and clarifies concepts related to spurious activation of SIS and presents several analytical expressions for quantification of STR including multiple contributions. They develop a set of simplified equations for a few architectures, and compare results against the equations given by the PDS method and ISA TR84, concluding that the results are similar.

1.6.4. Fault Tree Analysis

Fault Tree Analysis is well established as a very convenient method for dependability modelling. In comparison with other methods, it permits modelling of large and complex systems more easily. It provides a graphic representation to visualize the failure process and its basic causes, and it is easy to quantify. Based on these advantages, and the comparative account made in the previous sections, FTA has been selected to be used in this thesis, together with some additional techniques, for quantification of system dependability.

A fault tree is a graphic analytical model that represents the interrelationship between a potential critical event and the combination of events that cause that event. The main critical event being studied, which usually is an accident or a system failure, is called the top event of the fault tree. Fault tree analysis is a deductive top-down method: The immediate causal events of the top event are identified and represented as branches below it, then the immediate sub-causal events are branched below, and the procedure continues until the most basic causes are found. These are called the basic events. The causal events are interrelated by logic gates.

Once the fault tree has been constructed a qualitative and quantitative analysis are executed: The qualitative analysis of the fault tree intends to visualize paths of propagation of failure, and to identify the most critical events and weaknesses. When the fault tree is very small and simple this can be done just by direct analysis. Otherwise, the analysis requires reducing the tree to a *logically equivalent form, showing the specific combinations of basic events sufficient to cause*

the top event (Leveson, 1995). This is the identification of minimal cut sets (CS). A CS is a set of basic events that combined ensure the top event occurs. When a cut set cannot be further reduced it is called a minimal cut set (MCS). The order of a cut set is the number of basic events that compose it. Identification of the MCS permits visualization of which combinations of basic events lead to the immediate occurrence of the top events. Therefore, low-order cut sets, e.g. first and second order ones, are the most influential combinations in system failure. MCS reduction for small fault trees can be produced by hand, and computer algorithms are available for more complex situations (see Andrews & Moss, 2002). The quantitative analysis permits calculation of the likelihood of occurrence of the top event, based on the likelihood of the basic events. This can be expressed as a probability or frequency. The top event probability can be quantified using several methods (Andrews & Moss, 2002): gate-to-gate quantification, minimal cut sets and computer codes.

There are some other developments in fault tree analysis worthy of discussion. The most relevant to this thesis are:

1. House events. House events were originally named external events (Vesely et al., 1981), which are basic events that are not themselves faults and that can either occur or not to occur. Thus, they take a logical value of zero or one. Andrews (1994) proposed to use them to permit to switch on and off several branches of the tree in order to accommodate changing designs (such as needed for design optimization).
2. Modularization of fault trees. A technique used for splitting the fault tree into independent sub-trees and solving them separately (Chatterjee, 1975). This permits the solution of independent sub-structures of the tree by another method. For example, the solution can use Markov models in order to integrate time-dependencies (Gulati & Dugan, 1997). Then the probabilistic result is introduced in the fault tree as a basic event.
3. A similar idea is used by Signoret et al. (2007), who introduce time-dependent analytic models into the basic events of the fault trees (so they are empowered to handle time dependencies). Dudit et al. (2008) uses Markov models or Petri Nets for solving sub-trees.

Notice that an alternative way of empowering fault trees for handling time dependencies, so to avoid using MA, is modularization. A time-dependent analytic model could be inserted instead of including Markov models in the modularized sub-trees. The entire fault tree can be then evaluated a repeated number of times and then averaged. This method is proposed in Chapter 5.

An additional method for reduction of fault trees from a generic form to a particular design has been proposed by Lu & Lutz (2002). They developed a recursive algorithm that permits a top-down pruning of fault trees. This is intended to enable reuse of a generic Fault Contribution

Tree (FCT) that represents an entire product family, and which sub-trees and branches represent commonalities and variabilities respectively, for modelling a specific member of the family (i.e. a product). This is achieved by identifying the specific features of the product (i.e. its commonalities and variabilities), and then cutting down (pruning) the product family FCT iteratively in a top-down order until achieving a satisfactory representation of the product according to the resolution needed. The method has been further developed for construction of Software FTA for product lines in Dehlinger & Lutz (2004, 2006). This method has not been explored here, but it has been proposed as an interesting opportunity for future research in Chapter 7.

1.7. LIFECYCLE COST MODELLING

Lifecycle costing is fundamental for the successful implementation of a safety system. Achieving plant safety is a noble goal itself, but at the end of the day the safety system must be cost-effective; otherwise the burden of system cost could be so high that the plant owner could choose rather to withdraw from the enterprise of putting the plant into operation. This is recognized in the ALARP principle, which establishes that, after demonstrating that all the risk reduction measures considered “reasonable” have been put on place to lower the risk level, the effort (usually expressed in monetary terms) needed for further reduction through the implementation of additional risk reduction measures is not “*grossly disproportionate to the benefit to be gained*” (HID, 2003), and that the residual risk is not disproportionate to the benefits to be obtained. This is usually made through a cost-benefit analysis.

There are a few Lifecycle Cost models (LCC) developed specifically for safety instrumented systems, some of them specific for process industry. The choice of model is mainly based on its capacity to include all relevant factors and the availability of data to apply it. A comprehensive guide to lifecycle costing can be found in Dhillon (1989), which also deals with reliability applications. Maybe the first model developed specially for process safety systems was provided by SINTEF (Lydersen & Aaro, 1989). This model was developed as part of the work for the PDS Method (the latest version of which is Hauge et al., 2006a, and which is based on the standard IEC (1987)). They define the LLC of a system as “*the total cost to the user of the purchase and installation, and the use and maintenance during a stated period of life*”. It therefore must include not only the initial acquisition cost, but operation costs as well. Their general model is:

$$LCC = LAC + LSC + LUC \quad (1.27)$$

Where:

LAC = Life Acquisition Cost. This includes the cost of the initial investment: equipment cost plus, design, installation and commissioning.

LSC = Life Support Cost. It comprises resources for operation and maintenance, including their yearly cost during the entire operational life.

LUC = Life Unavailability Cost. The loss of production (STR) is included here. However, the expected costs due to potential accidents caused by hazardous situations; i.e. cost of safety unavailability (i.e. failure to prevent accidents) is not included here.

To include accident cost in the quantification of LUC is a controversial and difficult issue. This may be the reason for it not being included in the model of Lydersen & Aaro (1989). The author considers, however, that it is a significant omission, because the real benefits of risk reduction cannot be quantified and included in the LCC. Notice that Lydersen & Aaro also used the present value of yearly costs to transform the yearly cash flow to present value of yearly costs: i.e. annuities.

The NORSOK¹⁰ standard O-CR-001 (NORSOK, 1996) intended to standardized LCC calculations for systems and equipment. It also takes into account the time value of financial costs. Its main factors are capital cost, operating cost and cost of deferred production. The latter is calculated based on the average number of critical failures per year. This can be adjusted to include spurious trips and dangerous failures, but for quantifying only the losses in production. This standard advises to include the uncertainty of calculations (standard deviation), based on the confidence of the input data.

An alternative method, specific for SIS, is given by Goble (1998). He splits the primary costs categories into two: procurement and operating costs (Eq. (1.28)). The relationship of these two cost factors and equipment reliability is given in Figure 1.9.

$$LCC = C_{\text{PRO}} + C_{\text{OP}} \quad (1.28)$$

Procurement cost includes design, purchase, installation and start-up. Operation cost basically includes cost of engineering changes, consumables, fixed maintenance costs and cost of failure (risk costs). These costs are treated considering the time value of money, which is discussed below with the complete LCC model.

¹⁰ NORSOK stands for *The Competitive Standing of the Norwegian Offshore Sector* (in Norwegian), a series of standards issued by the Norwegian Technology Standards Institution (NTI)

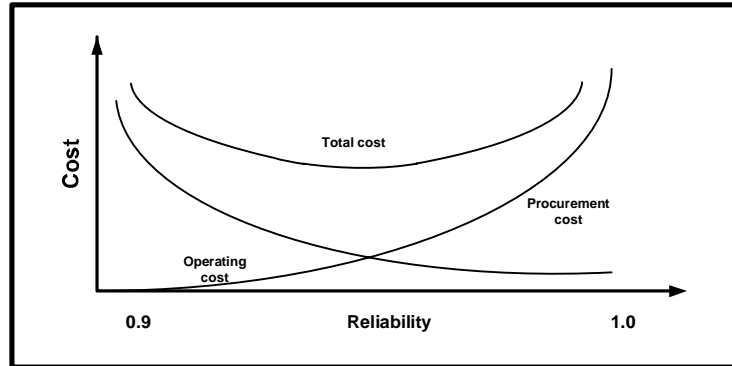


Figure 1.9. Cost behaviour against reliability (Goble, 1988)

Goble affirms that “a SIS represents a special case, as failure costs consist primarily of risk costs”, and that this should be considered in lifecycle cost analysis. He proposes to calculate the cost of risk based on the multiplication of the probability of the event without the SIS times the PFD of the SIS:

$$P_{\text{EVENT with SIS}} = P_{\text{EVENT without SIS}} \times \text{PFD}_{\text{SIS}} \quad (1.29)$$

This event is clearly a hazardous one, i.e. a potential accident, and thus includes the risk’s cost by dangerous failures. This idea has been included in the LCC model presented in this thesis. As with SINTEF, Goble compels the reader to include the cost of false trips.

Kawauchi & Rausand (1999) presented a report with a survey and comprehensive analysis of lifecycle costing for the oil and chemical process industry. They did not present any specific mathematical model, but rather an exposition of the theory behind the LLC.

Studies for the nuclear industry have many times been ahead of the process industry. Martorell et al. (2002, 2005a) introduced an important contribution to the models for system operation cost considering the test and maintenance strategy and the technical specifications in nuclear power plants. Martorell et al. (2002) developed yearly cost functions test, preventive maintenance and corrective maintenance costs, specifically applied for safety system optimization. It also included outage cost and the cost of system overhaul. The cost of outage time c_u (a concept similar to shutdown) is also included, which considers loss of production:

$$c_u = T_s \cdot c_{hs} \quad (1.30)$$

Where T_s is the start-up time and c_{hs} is the cost of production per hour. Martorell et al. (2005a) subsequently firmly established the cost factor as an additional objective for optimization of

safety systems, formulating the system objectives as RAMS+C. They also added to the operation cost model the cost of possible accidents c_a :

$$c_a(\mathbf{x})=R(\mathbf{x})\cdot c_{acc} \quad (1.31)$$

Where $R(x)$ is the yearly risk contribution and c_{acc} is the cost per accident. Eqs. (1.30-1.31) account therefore for equivalents of cost of spurious trips and dangerous failures, which is included in the risk cost in the LCC model used in this thesis (presented in Chapters 3-4).

1.8. SAFETY INSTRUMENTED SYSTEM OPTIMIZATION ISSUES

This chapter has presented the introductory theoretical material to understand and analyze Safety instrumented Systems, the theory and approaches for modelling them and the normative framework currently valid. IEC 61508 has progressively become the international standard for safety related systems in several different industrial sectors. It fundamentally regulates the ensurance of system functional safety. However, other works, mainly the PDS method, have put emphasis on including the analysis of loss of production as a consequence of spurious activations, so a proper balance between safety and reliability is achieved. This should be evaluated against the system lifecycle cost. A convenient balance between these three objectives can be achieved by two strategies: optimizing the design of the system and optimizing maintenance, specifically the testing policy. This is the problem approached in this thesis.

Design of safety systems relies basically in either choice of system components according to their reliability specifications or the allocation of redundancy. Component choice depends largely on the technological options available. Redundancy allocation may improve safety but affect reliability, or vice versa. It also has a considerable impact on system cost. In addition, the increment in subsystem redundancy using identical components conveys an inherent problem that can significantly negate its benefits: the simultaneous failure of all the redundant components due to the same cause, called Common Cause Failure. One of the most effective techniques for counteracting CCF is to implement redundancy using technologically diverse components. Increasing redundancy, however, can improve one dependability objective but can affect the other. This can be counteracted using partial redundancy, implemented by voting architectures (MooN systems).

Testing is an activity that can largely benefit system performance without actually modifying its design. It requires the choice of optimal policies, which includes the testing frequency and its strategy. Too much testing can however convey negative effects, like large system downtimes and event transients that may increase spurious trips. This impacts negatively the Lifecycle Cost

and the confidence that the operator places on the safety system. To find a correct balance in testing policies is, therefore, another optimization problem. The way testing policies relate to system design changes when Moon voting architectures are introduced. This may increase the benefits of both testing and design, but also may increase the complexity of the optimization problem.

It is when considering these opportunities and challenges that the analysis and research of the safety system optimization problem becomes important. Those described above are the opportunities for research that are explored in this thesis. This work is mainly focussed on the safety systems used in the process and oil and gas industries, but its techniques, methods and results are sufficiently general to be easily extrapolated and applied to other sectors; e.g. the nuclear industry, transport, aerospace, etc.

CHAPTER 2

RAMS+C optimization and Genetic Algorithms

The previous chapter provided an introduction to Safety Instrumented Systems and the theory behind them. It also established that the metrics for measuring the dependability performance of the SIS are the average Probability of Failure on Demand (PFD_{avg}) and Spurious Trip Rate (STR), which will be used for measuring system safety and reliability respectively. These two dependability metrics, together with the system Lifecycle Cost (LCC) are the three objectives to optimize, and are encompassed within the generic concept of RAMS+C (Reliability, Availability, Maintainability and Safety plus Cost). The problem to solve becomes therefore a RAMS+C multi-objective optimization. This chapter gives an introduction to the theory and techniques of the approach adopted for the solution of this problem.

Primarily, an introduction to the multi-objective optimization problem, together with the basics for its solution, is presented. Several classical optimization methods are outlined, so a comparison can be drawn against the approach chosen in this work. In order to motivate the interest of the reader, a review of the most influential works of RAMS optimization with Genetic Algorithms (GAs) is given before introducing their theory. The principles of GAs are subsequently explained. Finally, the two specific GAs used for the optimization cases of the thesis are detailed.

2.1. THE MULTI-OBJECTIVE OPTIMIZATION PROBLEM

2.1.1. The general problem

Optimization is the process of selecting the values of the input variables of a problem in order to find the optimal solution. The input variables are those that can be controlled or modified by the optimizer. The optimization seeks to either minimize or maximize some specific objectives of the problem at hand, which are the output of the problem. Each objective is defined by an objective function.

A single-objective optimization is the process where only one objective is optimized. The optimizer in this case gives as a result one single optimal solution. The case of multi-objective optimization is different. Multi-objective optimization deals with two or more objectives to be optimized simultaneously. The different objectives can present different relationships between

one another: independency, conflict or harmony. It is very common to find that several of the competing objectives present conflictive relationships. Therefore, it is not possible to find an ideal solution that is best for all the objectives. There is no single optimal solution but a group of several trade-off solutions that are all optimal. Without additional preference information, no single solution can be said to be better than another. This is what the multi-objective optimization problem addresses.

The problem of multi-objective optimization is one of simultaneously minimizing the n different objectives of an objective function vector $\mathbf{f}(\mathbf{x})$:

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_n(\mathbf{x})) \in \mathbf{Y} \quad (2.1)$$

Where \mathbf{x} is a decision variables vector

$$\mathbf{x} = \{x_1, x_2, \dots, x_n\} \in \mathbf{X} \quad (2.2)$$

Usually the objective space \mathbf{Y} is restricted by a vector of constraints; e.g.:

$$\mathbf{g}(\mathbf{x}) = (f_1(\mathbf{x}) \leq g_1, f_2(\mathbf{x}) \leq g_2, \dots, f_n(\mathbf{x}) \leq g_n) \quad (2.3)$$

The vector of constraints has the objective of forcing the optimizer to provide only feasible solutions (those that are useful for the decision maker). As mentioned by Marseguerra et al. (2006), the vector of constraints place limits on the objective space, considered as implicit constraints. Alternatively, limits can be imposed directly on the decision variables vector, which are called explicit constraints. This is the case when it is intended for the decision variables varying only between realistic limits.

As mentioned above, it is generally impossible to find a unique solution that optimizes all the objectives at once. In its place, a set of non-dominated optimal solutions is found, which contain the best solutions in decision space \mathbf{X} , called the Pareto-optimal set, which constitutes the Pareto front. This implies that a single solution is to be picked up from this set of alternative solutions by the decision maker, which usually is a human.

The bounds placed on the decision variables vector constitute the decision variables space. Since the decision variables vector has correspondence with an objective function vector, there is a mapping between the decision space (where the search is performed) and the objective space: there is one solution in the objective space corresponding to each point of the decision space. The bounds on the objective space, being those imposed by the limits of the variables or the constraints vector, constitute the feasible region of the search space. Figure 2.1 shows a representation of the mapping between the decision space and the objective space.

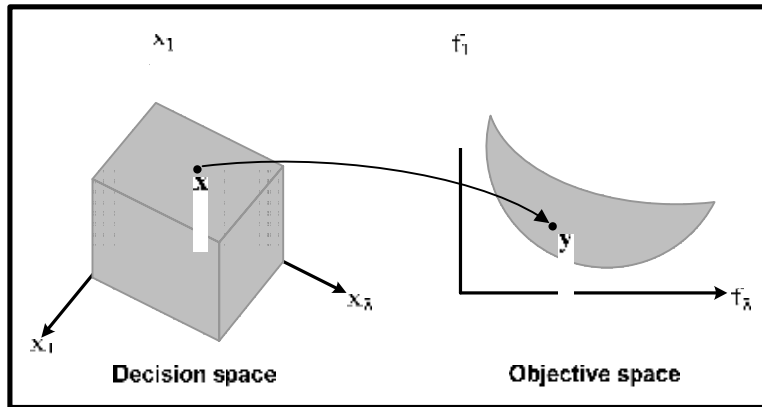


Figure 2.1. Representation of the decision and the objective spaces

2.1.2. The multi-objective optimization problem for Safety Instrumented Systems

The optimization of SIS falls within the category of reliability optimization. A complete analysis and classification of the reliability optimization problems can be found in Kuo et al. (2001). Several different optimization cases are developed in this work. They can be summarized into the following general cases:

- Optimization of design with redundancy and reliability allocation. System design is improved choosing the level of redundancy in subsystems, and choosing from specific available components that have different reliability specifications (technological principles, failure rates, diagnostic coverage, etc.). Therefore, the reliability values available are discrete values. There are some studies in which the reliability allocation is continuous (e.g. Salazar et al., 2006). This is, however, hardly applicable to real-life cases, where components have to be selected from *off-the-shelf* options. This limits the range of choices to some few discrete cases. In the optimization, each technological choice is designated with an integer index number. These index numbers are used as decision variables to indicate the type of component chosen.
- Optimization of proof testing policies. The design of the system is fixed, and the system test frequency and strategy are optimized. The test frequency, determined by the Test Interval (TI) is indicated in hours. The test strategy establishes how the tests of the redundant components are scheduled with respect to one another. This is set with the time of first test of each i^{th} component T_{Pi} (which is repeated every Test Interval). Thus, both the TI and T_{Pi} can be expressed as integer variables.

Given that the decision variables in both generic problems are integers, they can be viewed as non-linear integer programming problems. These problems are NP-hard, which are usually unlikely to have “*computationally efficient algorithms for exact optimal solutions*” (Kuo et al., 2001). There are no powerful generic algorithms for solving non-linear problems.

In general, the problem is to minimize the objectives: Average Probability of Failure on Demand, Spurious Trip Rate and Life Cycle Cost:

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) = (PFD_{avg}(\mathbf{x}), STR(\mathbf{x}), LCC(\mathbf{x})) \quad (2.4)$$

Where \mathbf{x} is a vector of integers,

$$\mathbf{x} = \{x_1, x_2, \dots, x_n\} \quad (2.5)$$

Subject to:

$$l_i \leq x_i \leq u_i \quad \text{for } i = 1, \dots, n$$

The problem is combinatorial (choice of components or test times) and stochastic (component failure rates, dependability outputs) (Kuo et al., 2001). It is therefore a discrete optimization.

2.1.3. Pareto dominance and optimality

The criteria for determination of which are the optimal solutions from all the solutions in the feasible space rest on the concept of Pareto dominance and optimality. These permit comparison of any two solutions from the search space with respect to their multiple objectives and determine which are the optimal. According to Deb (2001) a solution **A** dominates another solution **B** if:

- The solution **A** is not worse than **B** in all objectives, and
- The solution **A** better than **B** in at least one objective

For the minimization case, this is:

$$f_i(\mathbf{A}) \leq f_i(\mathbf{B}) \quad \text{for all } i \in \{1 \dots n\}, \text{ and}$$

$$f_i(\mathbf{A}) < f_i(\mathbf{B}) \quad \text{for at least one } i \in \{1 \dots n\}$$

It is therefore said that:

$$\mathbf{A} \prec \mathbf{B} \quad (2.6)$$

If solution **A** dominates **B**, **A** is better than **B**. A solution that is not dominated by any other solution is a Pareto-optimal solution. The entire group of Pareto-optimal solutions form a Pareto-optimal set, and the curve they form when being joined is called the Pareto-optimal front

(Deb, 2001). The size of the entire Pareto-optimal set is usually too big to be completely determined. Therefore, the optimizer must seek to find an optimal set of solutions that represents it as faithfully as possible.

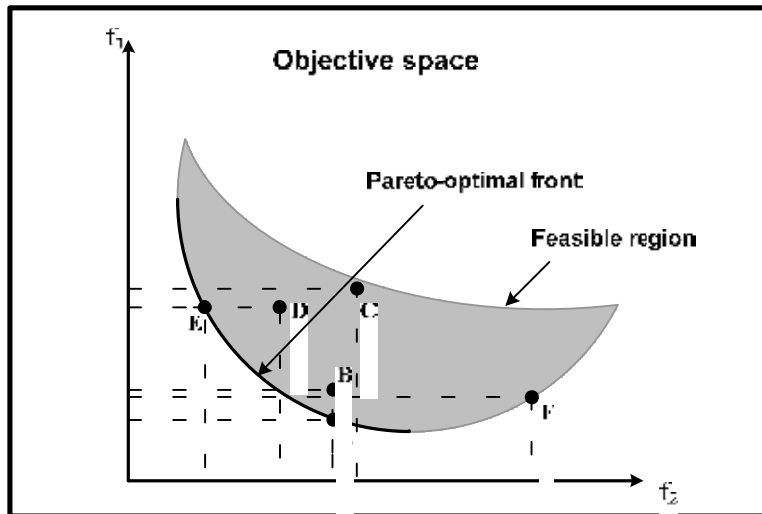


Figure 2.2. A six-solution population, demonstrating the concept of dominance

Figure 2.2 illustrates the concepts explained, showing a population of six solutions. Solution **A** and **E** are Pareto-optimal solutions because they are non-dominated by any other solution of the set. Observe that solution **B** has the same value in the objective f_2 that **A**, but **A** is better in f_1 than **B**; thus **A** dominates **B**. The same situation can be observed with solutions **E** and **D**. Solution **C** is clearly a non-optimal one, because it is dominated by many other solutions. Also notice that although solution **F** is at the border of the feasible region, it is not an optimal solution because it is dominated by solution **A**, which is better in both objectives. Notice that the solutions contained in the Pareto-optimal front are the best achievable trade-offs of the objectives of the problem.

It is important to highlight the comment made by Deb (2001) that “*there exist multiple Pareto-optimal solutions in a problem only if the objectives are conflicting to each other*”. Otherwise, if they are not conflict one single optimal solution is achievable.

An optimizer must meet two basic requirements:

- Proximity. The obtained best-known Pareto front should be as close a possible to the true Pareto front (Konak et al., 2006).
- Diversity. The obtained Pareto set must be uniformly distributed and cover all areas along the Pareto front, so that it is a true representative sample of it.

Purshouse (2003) formulated the additional objective of pertinence, meaning that the obtained Pareto set must provide solutions on the pertinent regions of interest. This, however, implies to guide the search based on previously expressed preferences.

2.2. MULTI-OBJECTIVE OPTIMIZATION TECHNIQUES

2.2.1. Treating the multi-objective problem as a single-objective problem.

The multi-objective optimization problem is a vector optimization problem. There are traditional methods of dealing with this. One is to optimize a single objective function composed of the sum of all normalized objectives multiplied each by a user-supplied weight, called the Weighted Sum Methods (Deb, 2001). The weights are chosen according to the relative importance of each objective. However, as an alternative, all the objectives can be combined into a single function to optimize using weighted metrics (Weighted Metric Method). Another approach is to optimize only one objective at a time while the other objectives are used as constraints included into an ε -vector. This is called the ε -Constraint Method. This method *“often lead to a solution which may not be the best or most satisfactory”* (Kuo et al., 2001). To formulate the weights for each objective or the weighted metrics requires knowledge of the process and expression of previous preferences on the objectives. The same happens in the ε -Constraint Method, since one of the objectives must be prioritized over the others, and previous knowledge of the desired boundaries of the objectives used as constraints is needed. The required normalization of objectives in these methods necessitates previous knowledge of minimum and maximum values of each objective.

Goal Programming methods intend to find solutions towards a predefined target for one or more of the objectives. In case the solution is not reachable, the method seeks to minimize the deviations from the target. The methods can use weighted factors for deviations (Weighted Goal Programming) and minimize the weighted sum of deviations (which in fact becomes a single-objective optimization), or to categorize the goals with priority ordering (Lexicographic Goal Programming) and solve then in sequence.

Deb et al. (2001) finds several difficulties with these classical methods. First of all, they convert the multi-objective optimization problem into a single-objective problem. This must be solved using a single-objective optimizer. In order to find another Pareto-optimal solution, the parameters must be changed and a new run performed. There is no guarantee of finding a solution in the entire Pareto-optimal front. They therefore provide only one single Pareto-optimal solution per run, and the algorithms require good previous knowledge of the problem.

The main drawback the author finds in the methods is the fact that the objectives must be prioritized one over the others, and this prioritization be expressed as quantities. This can be a complex task. It may also be quite subjective; solely left to the programmer's criteria. It is true that for SIS safety there must be a prioritisation. However, the author considers that better balanced objectives may be found if all the objectives have the same importance in the optimization and are treated independently. The *a priori* articulation of preferences may be useful, but only if it is based on a deep knowledge of the problem at hand. It is also the case that the constrained bounds of the objectives are not always previously known. To obtain as a result only one or few Pareto-optimal solutions could conceal some other good optimal solutions from the decision maker. In addition, as Kuo et al. (2001) mentions, the Pareto-optimal solutions given by the traditional methods mentioned above may be quite sensitive to the weights and relative importance given to the objectives. To leave this articulation to the decision maker *a posteriori*, visualizing all the options available on a set of well-distributed Pareto-optimal solutions in a more interactive decision making process, may provide better trade-offs.

2.2.2. Multi-objective optimization

A very comprehensive review of optimization methods in reliability is given by Kuo et al. (2001), highly based on Kuo & Rajendra-Prasad (2000). They chiefly report the main advances in the field from 1980 onwards. It is interesting to note that they state that very little attention has been paid to application of exact solution methodologies by researchers (from 1980) for reliability optimization. The methods presented fall within the four broad techniques of exact methods, approximation methods (based on non-linear programming), heuristics and meta-heuristics. In general, they have been applied for single-objective optimization problems.

Heuristics methods are usually bespoke methods applicable to the specific problem for which they were developed. An account of these methods applied to reliability optimization can be found in Kuo et al. (2001). A brief review of other methods (other than meta-heuristics) is made herein. Dynamic programming can be quite effective for certain deterministic and stochastic problems, and it is powerful for discrete optimization problems. These methods transform a n -variable optimization problem into a multi-stage decision making process. This transformation is, nevertheless, not always simple, and may require a lot of ingenuity. It is good for problems with no more than one constraint, since its computational complexity increases exponentially with the number of constraints. This is the main drawback of this method. It is usually useful for redundancy optimization if the system is series or hierarchical series-parallel. An advantage is that it gives an exact optimal solution for problems with integer data input (in single-objective optimization).

Several discrete optimization methods are available for solving non-linear integer programming problems; mainly dynamic programming, branch-and-bound techniques, implicit enumeration methods and lexicographic search. Nevertheless, their computational complexity is also very high. The structure of the problem, number of variables and number of constraints highly influence the performance of these methods. A singular advantage is that they give an exact optimal solution. However, in the opinion of Kuo et al. (2001), there is not a unique discrete optimization method that can be considered to be the best for all discrete reliability optimization cases.

Non-linear optimization problems with integer variables can be dealt with by solving its continuous version and rounding off the optimal values. These non-linear programming methods are based on three primary approaches: Gradient methods, Lagrange multipliers and solving unconstrained optimization problems by penalizations. Penalizations are widely used for their simplicity, but their convergence may be slow. These methods are useful even when functions do not satisfy assumptions of continuity and differentiability. By penalizations, there are two methods for solving an unconstrained optimization problem: derivative-type methods and search methods. The former require an iterative evaluation of partial derivatives of the objective function. Although they converge faster than search methods, if the number of variables is large they involve a lot of preparatory work. Thus, search methods may be more convenient for large scale problems. Generally, non-linear programming methods have high computational complexity. In addition, their main drawback is that rounding off may not give an optimal solution when solving discrete reliability optimization problems.

The methods described above are highly complex in terms of computation. Consider in addition that Kuo et al. (2001) mention them only in single-objective optimization applications for relative simple cases: Usually for only either redundancy or reliability allocation for reliability optimization and subject to reliability constraints. The complexity of implementing these methods for multi-objective optimization may be excessive.

Deb (2001) suggests that classical methods may face strong difficulties in solving practical optimization problems when major fix-ups are not put in place. He states that, generally, these methods are designed to solve a specific type of problem. In addition, a big drawback is that if they fall into local minima it is not possible to escape it. The finding of an optimal solution depends largely on the initial solution chosen. Many real world problems have discrete variables, with discontinuities and noisy search spaces, and classical methods are not very efficient in handling them.

Evolutionary Algorithms (EAs) are metaheuristic methods based on the evolutionary principle of survival of the fittest. They are part of soft computation artificial intelligence. They mimic the natural selection and evolution process based on stochastic search techniques. EAs rely on the process of selection and search of a population of potential solutions to evolve towards a set of optimal solutions. Perhaps the most popular and widely researched of the group of EAs are the Genetic Algorithms (GAs). GAs are explained in detail in Section 2.4, and two particular algorithms are discussed in Chapters 3 and 5. For a complete overview of multi-objective EAs and a good explanation of the basics of GAs the reader can see Deb (2001) and Haupt & Haupt (2004). A review from the point of view of RAMS optimization can be read in Marseguerra et al. (2006) and Konak et al. (2006).

As mentioned above, the problems of redundancy and reliability allocation, as well as testing policy optimization, is a stochastic non-linear integer programming problem. In addition, for highly complex problems the objective function may not be explicitly known. This is the case with changing system designs, where the system fault tree is dynamically rearranged to generate a new implicit objective function. For solving these problems dynamic programming may have dimensionality difficulties with increasing number of variables, and integer programming requires transformation of non-linear objectives into linear forms, which may be very complex. The problems are thus not easy to subject to analytical treatment. Discrete variables make it difficult to use gradient descent techniques and mathematical programming.

Genetic Algorithms (GA) present several features that make them advantageous over classical optimization methods, especially for solution of multi-objective optimization problems. GAs work on potential solutions codified into *chromosomes*. This permits easy adaptation of complicated problems, codifying them instead of defining them analytically. Working with a pool of multiple solutions and iteratively modifying it, the search is steered towards multiple directions in the search space, which increases the likelihood of providing a superior better-known Pareto-front. In case it falls into local minima, this also permits to escape it. Different from classical methods, the population-based strategy makes it possible to obtain multiple optimal solutions in a single run. Since the operators of GAs operate on stochastic principles, and do not use deterministic rules, they do not assume any particular structure of the problem being solved (Deb, 2001). This confers them their great flexibility for application to a large diversity of problems.

GAs are useful for solving problems that are difficult for direct mathematical treatment (Kuo et al., 2001). What is more, they can be applied when the objective function is not known explicitly (as it is the case of the function evaluated based on changing fault trees). They can

handle high dimensional, non-linear and discrete problems, with discontinuous functions. This is remarkably useful for solution of real-life problems. They permit implementation with few assumptions and constraints; avoiding unrealistic assumptions and approximations being made. Being population-based, GAs can handle many different design scenarios efficiently, which is the case of the design and test optimization problems treated in this work. Another big advantage is that GAs provide a pool of several optimal solutions, which conveys great flexibility for decision making.

Perhaps the big popularity of application of GAs is due to their flexibility and simplicity for being adapted to a broad range of different applications. As Coello-Coello (1999) concisely points out, “*they are less susceptible to the shape or continuity of the Pareto front, (which) are big issues for mathematical programming techniques*”. GAs are surely not the perfect method universally applicable. As the number of objectives and the complexity of their objective function increase, they can become quite computationally expensive. The main disadvantage can be the series of design parameters that the user has to choose or “tune” for the GA to work efficiently, tailored to the specific application in hand. Nevertheless, as said by Schwefel (2000) a universal method able to solve all problems effectively and efficiently cannot exist. GAs can still be robust to the lack of perfection in some of their parameters, and deliver results satisfactory enough for the decision making process to take place.

2.3. OPTIMIZATION OF RAMS+C WITH GENETIC ALGORITHMS

According to Kuo & Rajendra-Prasad (2000) the main structures approached in reliability optimization are:

- Parallel-series systems
- k -out-of- n : $G(F)$ systems
- Network systems
- Other unspecified systems

Regarding system design optimization, mainly three kinds of problems have been addressed:

1. Reliability Allocation (RelA). The reliability of the components or subsystems is changed in order to find an optimum. Depending on the steps of reliability variation, this can be continuous (CRelA; e.g. steps $\leq 10^{-6}$) or discrete component reliability allocation (DRelA). Discrete allocation corresponds to the selection of reliability values from a predetermined set of components, likely *off-the-shelf*, which is usually the real-life case.

2. Redundancy Allocation (RedA). Reliability is optimized selecting redundancy levels for the subsystems or components. When k -out-of- n architectures are included, optimization may also involve choice of the number of voting components.
3. Redundancy and (component) reliability allocation (Red&RelA). A mixture of the two previous cases.

A complete discussion of all reliability optimization models can be found in Kuo et al. (2001). Optimization of system Test and Maintenance (T&M), also treated as Surveillance and Maintenance (S&M), can be considered as a different problem category. An account of the main works in this area is included below, but it is discussed in more detail in Chapter 5.

A review of the development of optimization by Genetic Algorithms in system RAMS+C is given next. It is limited to the architectures most relevant to Safety Instrumented Systems (i.e. no networks or more complex configurations). Application of GAs to reliability optimization is fairly recent. Practically, this field of knowledge has been developed since the mid-nineties. Some surveys useful as a general overview are Kuo & Rajendra-Prasad (2000) for general system reliability optimization, and two surveys on GA-based approaches for reliability optimization made by Gen & Kim (1999) and Gen & Yun (2006). These are however not comprehensive reviews.

Perhaps the first propositions for using GAs in reliability optimization are Gen et al. (1993) as cited in Kuo et al. (2001) and Smith & Tate (1993) as cited in Coit & Smith, (1994). Ida & Gen (1994) (as cited by Kuo et al., 2001), proposed (a series-parallel) system reliability optimization by redundancy allocation of components with several failure modes by GAs. Failure modes belonged to two different classes: O and A. Class-O failures fail an entire subsystem by the failure of one single component, while class-A failures do it only if all subsystem's components fail.

Coit & Smith (1994) made an early application of a Neural Network as a reliability (objective function) estimator of k -out-of- n systems for a design problem (where components of different types could be mixed). The problem was of RedA & DRelA through component selection, treating it as a combinatorial problem. The case was a series-parallel system cost optimization plus a penalty for exceeding a reliability constraint.

Painton & Campbell (1995) solved a DRelA problem for a series-parallel system of a personal computer: Maximization of MTBF. The problem was combinatorial and stochastic, since the inputs were failure rates represented by distributions (rather than deterministic point values).

The variables were three choices per component: no improvement (actual design), and two levels of improvement, each with a characteristic failure distribution. The constraint was cost, and the goal was to achieve a specific MTBF improvement. The constraints were incorporated into the fitness assignment function.

Coit & Smith (1996a) solved a RedA problem with choice of different components (DRelA) of a series-parallel system, including k -out-of- n : G subsystem redundancy. Components of different types could be mixed. The component's reliabilities were deterministic (e.g. uncertainty is not considered). The coding was made in integer values rather than binary. Fitness allocation includes a dynamic penalty function. Two examples were solved: Reliability maximization given cost and weight constraints, and cost minimization given reliability and weight constraints. A further work was made for generating an adaptive penalty for reliability optimization with cost and weight constraints and tested in several benchmark problems (Coit & Smith, 1996b). In Coit & Smith (1996c) they applied a Neural Network (in optimization by GA) for estimation of system reliability in a redundancy allocation problem with k -out-of- n redundancies.

Muñoz et al. (1997) made an early exploration of constrained optimization by GAs of Surveillance and Maintenance (S&M) of components with respect to risk (unavailability) and cost. The case study was for S&M intervals for a motor driven pump. The single objective cases address optimization of risk with cost constraints and vice versa.

Yang et al. (1999) approached the RelA problem of different systems composing a typical Pressurized Water Reactor (PWR) of a Nuclear Power Plant (NPP). Reliability allocation was based on probabilistic safety models (using event trees for probability of plant damage states (the hazards) and fault trees for system unavailability). They paid special attention to the formulation of a realistic objective function (based on value impact analysis) that considers the impact of potential hazards on the costs by health and investment risks. Optimization was made by improving or degrading the unavailability of the subsystems. The goal was to minimize overall cost, while preserving certain safety goals (constraints). They presented a form of parallel-coordinates plots to show the result of availability in different subsystems under different constraints.

Pattison & Andrews (1999) introduced the application of GAs for optimal safety system design that can be considered to be a pioneer application for design of a SIS. The optimization case is for a High Integrity Protection System (HIPS), which includes an Emergency Shutdown System (ESD). Redundancy, component choice, one MooN voting and test intervals are the variables.

The objective function is not given explicitly, and here the flexibility of GAs is well exploited. Instead, fault trees with house events are used. The resulting fault trees are solved by Binary Decision Diagrams (BDD), a methodology previously developed in Andrews (1994). The problem is RedA & DRelA (component choice). The fitness function is a sum of unavailability and penalization by exceeding the unavailability, cost, Mean Down Time (MDT) and STR.

Cantoni et al. (2000) applied a GA with Monte Carlo simulation for optimal plant design (for a case that could not be put explicitly in analytical form). The problem at hand was a series-parallel system with k -out-of- n :G. It included DRedA with selection of components with different failure and repair rates and cost. GAs were used for evaluation of the optimal given designs (i.e. validation). The optimization was made to an objective function of gains (profits vs costs), which included safety aspects: Cost of different types of accidents in the damage costs component. The methodology was then applied for optimizing maintenance and repair policies in Marseguerra & Zio (2000). The optimization was of maintenance intervals and number of repair teams. A profit function was used for the fitness function which included the cost of an accident having as a consequence of damage to the environment.

Martorell et al. (2000) integrated the Probabilistic Risk Analysis (PRA) modelling with a customized Steady State Genetic Algorithm (SSGA) for optimization of safety systems in NPPs. The application case deals with Test Intervals (TIs) variation of the components of a High-Pressure Injection System (HPIS) of a Pressurized Water Reactor (PWR) of a NPP. The optimization was made using an integer-coded GA. Analytical models were presented for the risk function (in terms of unavailability) and cost (of maintenance and test). Two optimization cases were presented: Optimization of risk under cost constraints and vice versa. Penalty functions were fully developed for the application cases. The overall concept was further developed in Martorell et al. (2002) to include maintenance intervals and Allowed Outage Times (AOT), besides the TIs, as decision variables. The Lifecycle Cost (LCC) model was also refined.

Giuggioli-Busacca et al. (2001) presented their first application of multi-objective GAs for reliability optimization. Optimization was performed for two examples. The first being a simple redundancy allocation with component choice. The problem was a two-objective maximization case: reliability during the mission time and revenues. The profit function included penalties for shutdown time. The second application was about optimization of TIs for the equipment items of a HPIS. Three objectives were optimized: availability, cost and exposure time (to radiation). The authors validated the algorithm against a single-objective solution. The cost function included cost of accidents. A MOGA is used, but does not specify if it the Fonseca & Fleming

(1993) algorithm.

Elegbede & Adjallah (2003) solved a multi-objective problem using a weighting technique, making it a single-objective optimization problem: Maximization of availability and cost (by availability performance) with penalties for a repairable series-parallel system. The decision variables were redundancy per subsystem, failure rate and repair rate. This means that the problem at hand had both continuous and discrete variables. The authors provided a contribution for tuning the algorithm for the specific application based on experimentation with some few runs of specific combinations.

Andrews & Bartlett (2003) extended the methodology of single objective optimization of Pattison & Andrews (1999) to a larger system. The application was for optimization of the probability of not functioning on demand of a Firewater Deluge System (FDS) plus (penalized) constraints of LCC (maintenance and test) and STR. The problem was of DRedA with component selection and test and maintenance intervals. The design was optimized for field instruments (transmitters, valves, pumps). Different failure modes: dormant and spurious are considered in the components data (failure rates and repair times) for the first time. The costs considered include only test and maintenance.

Greiner et al. (2003) compared the performance of various second-generation GAs (SPEA2, NSGA-II and NSGA-II with controlled elitism) in the multicriteria optimization of safety system design. They constructed fault trees and applied the “weight method” (published later in Gonzalez et al., 2004) to solve them. The problem was of component selection (discrete redundancy allocation) for a Containment Spray Injection System of NPP for optimization of unavailability versus cost. They found that the NSGA-II with controlled elitism gave the best overall average results.

Marseguerra et al. (2004a) explicitly included uncertainty in the parameters in a multi-objective optimization with GAs. One of the cases presented addressed optimization of surveillance intervals (STI) for a Residual Heat Removal Safety System of a Nuclear Boiling Water Reactor. The two objective functions evaluated the inverse of the s-expected system failure probability and the inverse of its variance. The uncertainties lay in the failure rate and duration of inspection parameters. Inclusion of uncertainty for surveillance requirements optimization has received further attention in more recent works, such as Martorell et al. (2008a). A survey on the topic can be found in Villanueva et al. (2008).

Marseguerra et al. (2004b) upgraded their method (Marseguerra & Zio, 2000) to multi-objective

GA with Monte Carlo simulation for the optimization of the Technical Specifications of a Reactor Protection Instrumentation System (RPIS) of a NPP, specifically regarding the testing policies (test intervals and allowable bypass times). The method starts by taking into account uncertainties in the model parameters to minimize both the mean value of the system unavailability and its variance (consequence of parameter uncertainty). A recent application of the methodology in Zio & Podofillini (2007a) includes system design (component selection) and spare parts allocation.

Martorell et al. (2004) introduced a general framework for multi-objective GA optimization of safety related systems based on the concept (coined by them) of RAMS+C criteria. They presented an application to the benchmark HPIS problem with optimization of maintenance based on the subset A[U]+C (unavailability and cost). It is a comprehensive case involving surveillance test, preventive maintenance and overhaul intervals, and allowed outage times for the valves and pumps of the HPIS. They contrasted the solution of the problem by two approaches: (1) direct multi-objective optimization and (2) converting the problem into several single-objective problems to be solved sequentially. It is also the second work to reportedly use a second generation GA: SPEA2.

In a subsequent paper, Martorell et al. (2005a) extended and refined the approach for a more comprehensive integration of Technical Specifications (surveillance requirements and limiting conditions of operation) and Maintenance (TSM) requirements. This with the objective of meeting the requirements of the nuclear industry standards. They introduced risk as function of RAM attributes. The costing model was here fully developed based on a previous work (Martorell et al., 2002), which includes reactor core damage as a risk scenario. The goals were included as constraints (of RAMS+C). The case study analysed was for an Emergency Diesel Generator. The objective function evaluated unavailability, incremental risk and cost.

Martorell et al. (2005b) explored the multi-objective optimization of the plant surveillance requirements simultaneously optimizing the TIs and the test planning (strategy), based on time-dependent modelling. Subsequently, Martorell et al. (2006) treated the HPIS system optimization with a double nested loop. An external loop optimized the TIs with a learning machine, while the internal loop (a multi-objective GA) optimized the test staggering.

Salazar et al. (2006) solved some benchmark problems using a second-generation multi-objective GA: NSGA-II. The problems addressed included RedA, RelA and RedA & RelA. Reliability was assumed to be discretized in steps of 10^{-6} , making practically continuous

reliability allocation cases with very big search spaces. Thus, they use a mixed real/integer codification. It validated the results against single-objective solutions of the same problems.

Borisevic & Bartlett (2007a, b) and Riauke & Bartlett (2008) upgraded the methodology presented in Pattison & Andrews (1999) and Andrews & Bartlett (2003) respectively for multi-objective optimization using the SPEA2 genetic algorithm. The HIPS and FDS problems were optimized this time having the system unavailability (with penalizations), cost and STR objectives (the last three were originally treated as constraints in the single-objective optimization studies).

Hussain & Todinov (2007) presented a single-objective optimization case of component selection for a series parallel system for minimization of total cost with reliability constraints, which basically included the initial investment for risk reduction plus the costs of system failure.

Zio & Podofillini (2007b) incorporate importance measures to the GA multi-objective optimization. The TIs of the HPIS were varied for optimization of unavailability, cost (of T&M) and an “*importance balance*” function. This last objective was based on the Fussell-Vesely measure of each component of the system, and intends to balance the testing frequency of each component according to its contribution to the other two objectives. In a subsequent work Podofillini & Zio (2008) compare the performance of the optimization using Birbaum and Risk Achievement Worth important measures against the Fussell-Vesely original proposition. These studies add interestingly to the Risk-Informed decision making strategy for safety systems, which is a concept presented in previous works by the same authors (an introduction to the topic can be see at Vesely & Apostolakis, 1999; and Vesely, 1999).

Taboada et al. (2007) approached an implementation form of the Decision Maker for reduction of the Pareto-optimal set by two different methods: Pseudo-ranking (similar to weighted sum method but with the objectives being ranked non-numerically by preference) and data mining clustering (where the choice is reduced to some few non-similar solutions, i.e. from individuals of different clusters), applied to several benchmark problems for demonstration. The case was of maximization of reliability and minimization of cost and weight by RedA with component selection using NSGA.

Tavakkoli-Moghaddam et al. (2008) proposed a single-objective optimization of plain reliability with cost and weight constraints, where the redundancy allocation and component selection redundancy scheme can be chosen between active and (cold) standby redundancies. A novel

encoding is made with matrix-formed individuals.

Some new hybrid approaches have emerged in the utilization of GAs for RAMS optimization. These are out of the scope of this work. However they are worthy of mention. Early applications are the optimization of the RedA problem with GAs and Neural Networks (NNs) for evaluation of the objective function, given by Coit & Smith (1994, 1996c). A brief account of some newer approaches can be found in Gen & Yun (2006). They propose some hybrid strategies of GAs mixed with other techniques to improve performance of the program and to solve problems inherent to GAs, such as premature convergence and the complicated, time consuming, tuning of the algorithm's parameters. Examples include GAs with Neural Networks for initialisation, GAs with local search (Hill climbing) and GAs with fuzzy logic. More recent works include Salazar et al. (2007), Rao et al. (2007) and Zio et al. (2008).

System modelling has recently been extended to Multi-state systems. In these, the system and its components have a range of performance levels, from perfect functioning to complete failure (Levitin et al., 1998). One example of optimization by redundancy and reliability allocation for series-parallel multi-state systems can be seen in Levitin et al. (1998). An application based on GAs was addressed by Tian & Zuo (2006). Multi-state systems are different from systems with two failure modes, and are out of the scope of this work. A comprehensive introduction to multi-state systems is given in Lisnianski & Levitin (2003).

Based on the review given above, a summary of the main developments in the subject is presented in Table 2.1. It can be seen that not much work has been developed in the field of SIS for the process industry. Although the level of system complexity has been growing, no study has addressed optimization towards compliance with IEC 61508, which implies some level of modelling detail (e.g. like several failure modes). Specific measures of SIS dependability, like quantifying system safety with the Probability of Failure on Demand and considering besides Spurious Trip Rate, have not been fully addressed. These last two points have been treated in the works of Pattison & Andrews (1999), Andrews & Bartlett (2003), Bosirevic & Bartlett (2007a, b) and Riauke & Bartlett (2008). Nevertheless, the requirements as specified by IEC 61508, such as system safety integrity, have not been approached by these works or any other optimization study so far.

Regarding the level of detail, especially Common Cause Failure and diagnostic coverage are issues that have been neglected. Another overlooked possibility is the integration of diverse redundancy for improvement of system performance. Coit & Smith (1994, 1996a) approached the option of diverse redundancy for optimization of plain reliability for simple systems. No

more development for practical safety systems have appeared. Just recently, Martorell et al. (2005b, 2006) integrated time-dependent modelling for more precise modelling of unavailability in optimization of test intervals and strategies, a matter of study with still a lot of possibilities. Finally, although some studies have included k -out-of- n architectures in design optimization (Coit & Smith, 1994, 1994a; Pattison & Andrews, 1999; Andres & Bartlett, 2003; Borisevic & Bartlett, 2007a, 2007b; Riauke & Bartlett, 2008), not much detailed work has been made for comprehensively studying its impact on the optimization of system design when fully-detailed modelling is made (i.e. IEC 61508), let alone with time-dependent models. Researchers that have addressed complex k -out-of- n architectures have done it with expensive methods, like Monte Carlo Simulation (e.g. Cantoni et al., 2000). All these issues deserve attention. These are opportunities for research that are explored in this work.

Table 2.1. Milestones in RAMS optimization with Genetic Algorithms

Year	References	Contributions	Remarks
1993-1994	[1] Gen et al. (1993) [2] Smith & Tate (1993) [3] Ida & Gen (1994)	First application of GAs	First applications of Genetic Algorithms to system RAMS has been developing since the first half nineties
1994-2000	[1] Ida & Gen, (1994) [2] Painton & Campbell (1995) [3] Yang et al. (1999) [4] Coit & Smith (1994) [5] Coit & Smith (1996a, 1996b) [6] Coit & Smith (1996c) [7] Pattison & Andrews (1999) [8] Cantoni et al., (2000)	Problems addressed in optimization of system design	GAs have been used for two main fields: system design optimization and optimization of test and maintenance. Regarding system design, some have addressed reliability allocation [2, 3] and redundancy allocation [1]. However, the most addressed problem is redundancy and reliability allocation; e.g. [4, 5, 7, 8]. The first hybrid applications of neural networks and GAs were made by Coit & Smith [4, 6].
1994-2003	[1] Coit & Smith (1994, 1996a, 1996c) [2] Patison & Andrews (1999) [3] Cantoni et al. (2000) [4] Andrews & Bartlett (2003)	Optimization of design with k -out-of- n structures	The most studied redundant structure is the parallel case. Nevertheless, the cited researchers have introduced k -out-of- n structures in the design.
1994-2004	[1] Coit & Smith (1994) [2] Painton & Campbell (1995) [3] Muñoz et al. (1997) [4] Yang et al. (1999) [5] Martorell et al. (2000) [6] Marseguerra et al. (2004)	Objectives used in the optimization	Since the unavailability of a safety system is critical, this is the most optimized objective [6], together with cost [1, 4]. Several authors use unavailability as a measure of risk [5]. In single objective optimization, constraints were placed as reliability [1], and cost [2, 3].
1997	[1] Muñoz et al. (1997) [2] Martorell et al. (2000, 2002, 2004, 2005a, 2005b) [3] Marseguerra et al. (2000, 2004a, 2004b) [4] Zio & Podofillini (2007a, 2007b)	Optimization of testing	The first application of surveillance (test) and maintenance interval optimization is [1]. Work in this area has been led by the group of Martorell [2] and Marseguerra and Zio [3, 4].
1999	[1] Yang et al. (1999)	Introduction of combinatorial methods	Introduced the solution of the objective function by combinatorial methods (i.e. fault trees), which has been developed in further works by other researchers.
1999-2005	[1] Yang et al. (1999) [2] Martorell et al. (2000, 2004, 2005a) [3] Giuggioli-Busacca et al. (2001) [4] Marseguerra et al. (2004a, 2004b)	Applications in the nuclear industry	Optimization applied to safety systems has mostly been developed in the nuclear industry, where the HIPS is a frequent case study.

Table 2.1. Milestones in RAMS optimization with Genetic Algorithms

Year	References	Contributions	Remarks
1999-2007	[1] Pattison & Andrews (1999) [2] Andrews & Bartlett (2003) [3] Borisevic & Bartlett (2007a, 2007b) [4] Riauke & Bartlett (2008)	Applications related to process industry	Applications relevant to the process industry are [1, 2]. They approach safety systems normally used in SIS and introduced optimization of more complex problems with design and test. Given the difficulty to define analytically an objective function in, they used fault trees with house events (synthesized by BDDs) instead of an explicit function. Ref. [2] introduced for first time the use of different failure modes (safe and dangerous) for quantification of system probability of failure and spurious trip rates. These works were upgraded to multi-objective optimization in [3, 4].
2000	[1] Cantoni et al. (2000) [2] Marseguerra & Zio, (2000) [3] Marseguerra et al. (2004b)	GAs and Monte Carlo simulation combined	To be able to model highly complex systems, [1] put together GAs and Monte Carlo simulation, a work that has been applied to other cases [2, 3].
2001	[1] Giuggioli-Busacca et al. (2001)	First application of multi-objective GAs	The first application of multi-objective GAs was made by Giuggioli-Busacca et al. (2001), in which objectives formerly treated as constraints were now included as objectives to optimize.
2001-2007	[1] Giuggioli-Busacca et al., (2001) [2] Martorell et al. (2005a) [3] Borisevic & Bartlett, (2007a, 2007b)	Number of objectives being optimized	In the most of the cases, only two objectives are optimized, being some few cases with three [1, 2] or four objectives [2, 3].
2004	[1]. Marseguerra et al. (2004a) [2]. Martorell et al. (2008a)	Inclusion of uncertainty	Marseguerra et al. (2004a) have included consideration of uncertainty in the parameters when using optimization by GAs. Other more recent works have developed it further [2].
2004-2005	[1] Martorell et al. (2004, 2005a)	Development of the concept of RAMS+C	Martorell et al. (2004, 2005a) introduced the broader concept of RAMS+C optimization, addressing technical specifications and maintenance of safety systems in NPPs. Their work intends to meet the requirements of nuclear industry regulations.
2005-2006	[1] Martorell et al. (2005b) [2] Martorell et al. (2006)	Use of time-dependent models for test & maintenance optimization	Time-dependent models were introduced by Martorell et al. [1] in multi-objective optimization of test & maintenance. They solve it with a double-nested loop in Martorell et al. [2], so that test intervals and test strategy can be both optimized.
2004-2007	[1] Marseguerra et al. (2004a, b) [2] Grainer et al. (2003) [3] Martorell et al. (2004) [4] Borisevic & Bartlett (2007a, b) [5] Riauke & Bartlett (2008) [6] Salazar et al. (2006)	Off-the-shelf GAs	Few authors indicate which GA they are using. [1] use a customized GA (which based on their references, could be thought to mix operators of the Fonseca & Fleming MOGA and SPEA). Grainer et al. [2] made the first application and comparison of second generation GAs (SPEA2 vs NSGA-II). Refs. [3, 4, 5] use SPEA2, while [6] made an application with NSGA-II for test in benchmark cases of reliability and cost optimization.
2006-2008	[1] Tian & Zuo, (2006) [2] Gen & Yun (2006) [3] Salazar et al. (2007) [4] Rao et al. (2007) [5] Zio & Podofillini (2007b) [6] Taboada et al. (2007) [7] Zio et al. (2008) [8] Tavakkoli-Moghaddam et al. (2008)	Latest developments	Latest developments have been attempting to get better balanced optimizations, incorporating importance measures into the objectives; e.g. [5]. The implementation of the decision maker is getting relevance as well, in the form of getting reduced Pareto sets [6]. The issue of complexity in the objective functions have led to mix other soft-computing approaches with Gas (i.e. hybrids), like Neural Networks and Fuzzy Logic [2, 3, 7]. Finally, applications to more complex cases are being investigated. For instance, Ref. [8] considers design with choice between active and passive redundancies. Another notable development in this sense is the application of GAs for optimization of multi-state systems [1].

2.4. PRINCIPLES OF GENETIC ALGORITHMS

Evolutionary Algorithms (EAs) is a group of techniques that mimic nature's selective process of evolution for solution of optimization problems. Genetic Algorithms are one of the most researched of EAs. They have been widely used for solution of single objective problems. For multi-objective problems, Genetic Algorithms have been extended to be able to handle two or more objectives, usually in conflict with each other.

2.4.1. Working principle

A Genetic Algorithm performs a stochastic search guided by the principle of natural selection based on individual genetics. The algorithm simulates the process of evolution of a population of individuals whose genetic characteristics are inherited from those ancestors that were fittest for survival, the same as the natural evolution of species does.

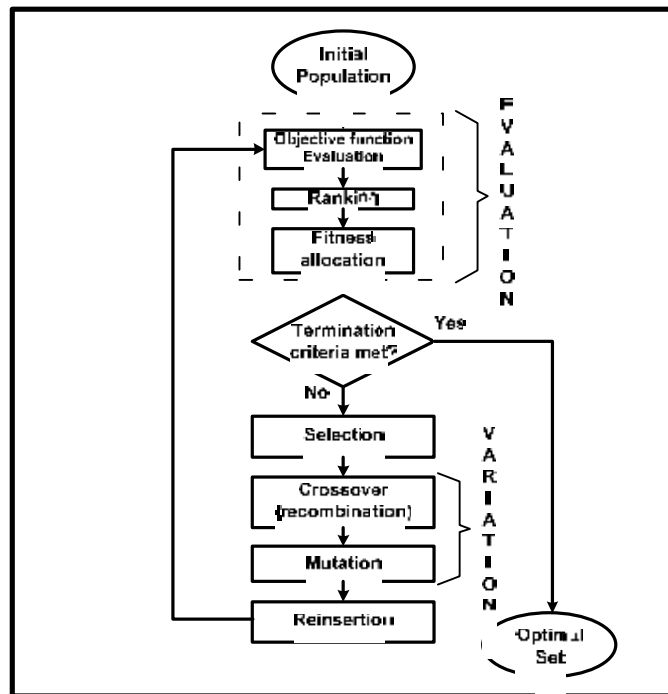


Fig. 2.3. Standard Genetic Algorithm flowchart

Figure 2.3 shows the general scheme of a GA. A GA works with a pool of solutions denominated *individuals*. They are evaluated, selected and mated to create new and hopefully better ones, which are fed into a new generation, making an iterative process that mimics the natural evolution. Each step of the GA is executed through the application of *genetic operators*. These operators together form the *breeding algorithm*. The standard GA follows generally the following steps:

1. **Creation of a random initial population.** The standard GA starts with a random *initial population* of several potential solutions in the decision space. Each *individual* is a coded representation of one set of decision variables (one single solution) in the decision space, called a *chromosome*. One chromosome is composed of individual discrete units or *genes*. This constitutes the *genotype* of the chromosome (see Figure 2.4). The codification is traditionally made in binary numbers, although nowadays other codes are used, such as integer and real numbers. In binary codification, each bit constitutes an *allele* (one gene is typically composed of more than one allele). These individuals pose a series of attributes that will determine their potential for being an optimal solution of the objectives of the problem at hand. Notice that since an individual is a potential solution to the optimization problem, many times they are referred to as a *solution* too.

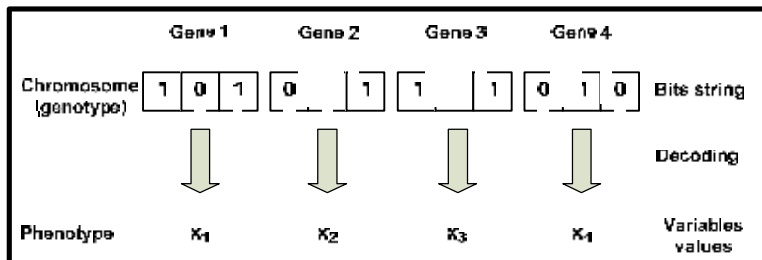


Figure 2.4. Encoding/decoding of an individual

2. **Evaluation of the individuals, which are ranked and fitness-assigned.** The chromosomes are decoded into the real values of the variables they represent. This is the *phenotype* of the individuals. With this the *objective functions* are evaluated. According to the evaluation in the objective space, the individuals are ranked and then assigned a *fitness* value, which determines their likelihood of reproduction in the next generation.
3. **Selection for reproduction (crossover), generating a number of new individuals.** A fraction of this population is selected for reproduction (recombination). The crossover is usually made by *mating* chosen pairs of *parent* solutions, generating a number of “children” called *offspring*.
4. **Mutation of the offspring.** The offspring are then mutated (commonly flipping one or more bits)
5. **Replacement of the offsprings into the parent population.** Offspring are reinserted in the population, usually substituting the same number of parents. This population represents a new *generation*.

The cycle of evaluation, selection, reproduction and reinsertion is repeated until a certain condition to stop the algorithm is met. This condition may be the exhaust of a generation count

or the compliance with a specific goal. At the end, the algorithm delivers a set of optimal solutions: the *Pareto-optimal front*. The individuals are evaluated, selected and mated to create new and hopefully better ones, which are fed into a new generation, making an iterative process that mimics the natural evolution.

2.4.2. Development of Genetic Algorithms

Genetic Algorithms (GA) were initially proposed by Holland (1975), while Evolution Strategies were firstly devised by Rechenberg (1973). These two fields have practically merged to form the basis of Evolutionary Algorithms. Comprehensive historical reviews of EAs are given by Coello-Coello (2006) and Adra (2007). This section is based on those works. The first implementation of a GA was the Vector Evaluated Genetic Algorithm VEGA (Schaffer, 1985). This GA was not based on Pareto-optimality. VEGA splits the population into subpopulations, each of them applied selection according to one of the objectives. Then they were merged back for crossover and mutation to get a general solution. The selection scheme of this approach was its main drawback, since it was unable to retain solutions with good trade-off performance, tending to keep ones that were very good in one specific objective (the *speciation* problem, according to Adra, 2007). Some other approaches were developed like Lexicographic Ordering GA.

The evolution of GAs up to date is categorized into two generations. Goldberg (1989) proposed the application of the concept of Pareto-optimality for ranking and selection in the GAs. This idea has been very influential in the development of the GAs after his publication, and the author considers this is maybe the main characteristic of the first generation of GAs. Goldberg also proposed to use niching to prevent the GA converging into a single point of the front (i.e. to enhance diversity). Several GAs were developed during the first generation. The most representative being the Non-dominated Sorting Genetic Algorithm NSGA (Srinivas & Deb, 1994), the Niche-Pareto Genetic Algorithm NPGA (Horn et al., 1994), and the Multi-Objective Genetic Algorithm MOGA (Fonseca & Fleming, 1993). The main difference among first-generation GAs is the mechanism of assignment of fitness (Adra, 2007). The Fonseca & Fleming MOGA is accepted as the most efficient algorithm of the first generation, based on a comparative study by Van Veldhuizen (1999).

The second generation is characterized by the incorporation of *elitism* to the GAs. They also have new and more sophisticated mechanisms of promotion of diversity. Elitism is a mechanism that helps to preserve good solutions during the optimization process. There are two different strategies for enforcing elitism (Konak et al., 2006): Storing the elitist solutions in a secondary archive, and maintaining elitist solutions in the population. The first implementation of elitism

was made in the Strength Pareto Evolutionary Algorithm SPEA (Zitzler & Thiele, 1999). This makes use of two populations, both participating in the optimization process. The second additional population keeps an archive of the non-dominated individuals, enforcing the elitist mechanism. Elitism with external population presents some drawbacks: It is computationally expensive as it requires management of the size of the second archive and an algorithm for reintroduction of elitism solutions into the population.

Other notable GAs of the second generation are the Pareto Archived Evolution Strategy PAES (Knowles & Corne, 2003), the Strength Pareto Evolutionary Algorithm 2 SPEA2 (Zitzler et al., 2002), an improvement of the SPEA, and the Non-dominated Sorting Genetic Algorithm II NSGA-II (Deb et al., 2000, 2002). The latter was presented as an improvement of the first-generation NSGA, but they can be considered different algorithms due to their significant differences. The NSGA-II, different from the other second-generation algorithms, does not use an external population for preservation of the best solutions. It instead combines the best individuals of both parent and offspring populations. This algorithm is in comparison so efficient, with such as good performance that it is considered *“a landmark against which other multi-objective evolutionary algorithms have to be compared”* (Coello-Coello, 2006).

In this work the two leading GAs, MOGA and NSGA-II, have been used for the optimization cases. They are described in Chapters 3 and 5 respectively. A complete historical review, describing the contribution of each GAs to the field, can be found in the two references given above.

2.4.3. The generic Genetic Algorithm

2.4.3.1. Initial population

The population of the GA is a group of N_{pop} individuals. The most popular *encoding* is binary, but real numbers are used too and sometimes integer codes. The choice of a particular codification is always problem-dependent. Choosing between binary and real operators, there are some differences between the genetic operators used, mainly for the crossover and mutation operators.

There is an argument for using Gray codes in binary representation. Gray codes have a Hamming distance of one between consecutive numbers. Hamming distance is the number of different bits between two strings of the same length. As cited by Haup & Haput (2004), some authors argue that binary codes may slow the convergence progress because the large variance that can occur in the crossover operator. In contrast, Gray codes reduce this variance and speed

convergence. However, Haupt & Haupt report as their experience that the conversion to Gray codes slows the GA and do not provide a significant improvement. As many of the tuning issues in GAs, the author believes the benefit of using Gray codes instead of binary must be problem-dependent.

The *population size* N_{pop} is one of the parameters of the GA to choose. The act of choosing the parameters of the GA is referred to as *tuning* the algorithm. As Marseguerra et al. (2006) discuss, a too small population can have as consequence low genetic diversity, causing the population to be dominated by similar chromosomes. This can cause premature convergence, convergence to a local optima, or lack of diversity in the found Pareto-optimal set. In contrast, a too large population can provoke an excessive genetic diversity, leading to the phenomenon of clustering around local optima. Mating of parents from a different cluster can create offspring that lack the good genetic qualities of both parents. These individuals are called *lethals*.

The initial population can be created following several strategies. Perhaps the most popular, for its simplicity, is to create it randomly. Other strategies can be followed in order to ensure diversity of guiding the search, such as creating per se a diverse population, or a population with individuals whose chromosomes favour one goal of the optimization.

2.4.3.2. Evaluation (fitness allocation)

Evaluation is a composite process. It requires evaluation of the objective functions based on the individual's phenotypes. This will give us the real values of each objective per individual. The second step is to allocate a (scaled) fitness value. The *fitness* is a measure that indicates the relative performance eligibility of a particular individual, and thus its ability to survive (Goldberg, 1989). A higher fitness indicates that the individual has a higher probability of surviving the next generation and of being chosen for reproduction.

A fitness function is used to assign the relative fitness of each individual. This is usually based on ranking; i.e. the relative rank of a particular individual among the population to which it belongs. It is therefore obvious that the Evaluation task comprises three steps: Evaluation of the objective function, ranking and fitness allocations (as indicated in Figure 2.3). Dominance-based ranking can be done by three different methods (Marseguerra et al., 2006):

- Dominance rank. A count of the number of solutions that dominate an individual. This is used in the MOGA.
- Dominance depth. An assessment of to which dominance front an individual belongs. Used in the NSGA-II.

- Dominance count. A count of the number of solutions and individual dominates. The method of SPEA2.

The fitness function is calculated based on the rank value (rather than on its actual objective function value, as Konak et al., 2006 indicate). In general, this chapter refers to the best fitness values as those of “higher fitness”. However, it is important to note that in minimization problems the fitness value may decrease as it improves. Thus, the lowest fitness values would represent the fittest individuals. It can be said that, in its most basic form, fitness is a function of ranking as follows:

$$f(\mathbf{x}, t) = r(\mathbf{x}, t) \quad (2.7)$$

Where f represents the fitness of the \mathbf{x} individual at generation t .

In addition, the fitness function can be modified in order to meet other requirements. For example, to add penalizations to individuals who violate established constraints. But most important is the adjustment of fitness to foster diversity in the population. Without additional measures to specifically promote diversity, the population may tend to form a few clusters of individuals (*genetic drift*), clearly affecting the performance of the algorithm. These measures are based on estimation of the density of the population in the neighbourhood of the individual being allocated its fitness. One example is *fitness sharing*, used in the Fonseca & Fleming MOGA, and *crowding distance*, the mechanism of the NSGA-II. A detailed account of these and the ranking methods is given in Chapters 3 and 5, where the MOGA and the NSGA-II are discussed respectively in depth.

2.4.3.3. Selection (for variation)

The selection of specific individuals for mating is based on their fitness value, which is equivalent to their reproduction expectation (Adra, 2007). Selection is a stochastic process. However, the better the fitness the higher the likelihood of being selected for reproduction. The selection operation consists on identification of good solutions and determination of their expected number of trials. This can be done by creating a *mating pool* (i.e. a temporary secondary parent population), substituting within it some bad solutions by multiple copies of the good ones. Once this is done, the selection is made.

There are several operators for selection. Some of the most popular are Tournament selection, Roulette Wheel Selection (RWS) and Stochastic Universal Selection (SRS). Tournament Selection (see Deb, 2001) is a simple algorithm that consists in playing tournaments between couples of individuals. The best individual (the one with better fitness) is chosen and placed in the mating pool. Each individual is chosen to play two tournaments. In this way, the mating

pool is filled with the winning solutions from the tournaments.

In the Roulette Wheel Selection (also known as Stochastic Sampling with Replacement SSR) an area of the roulette wheel is designated to each solution. This area is proportional to the solution's scaled fitness. The wheel is spun N times equal to the population size. One pointer indicates which solution is chosen every spin. In reality, when simulated in a computer, a "linear wheel" remains static and the pointer is moved by generating a random number r_s that indicates its position. The random number is generated in the range $[0, Sum]$, where Sum can be the sum of the expected selection probability (e.g. the cumulative probability=1). The selection probability of each individual is based on their fitness; and this determines its share of the wheel. This mechanism is illustrated in Figure 2.5.

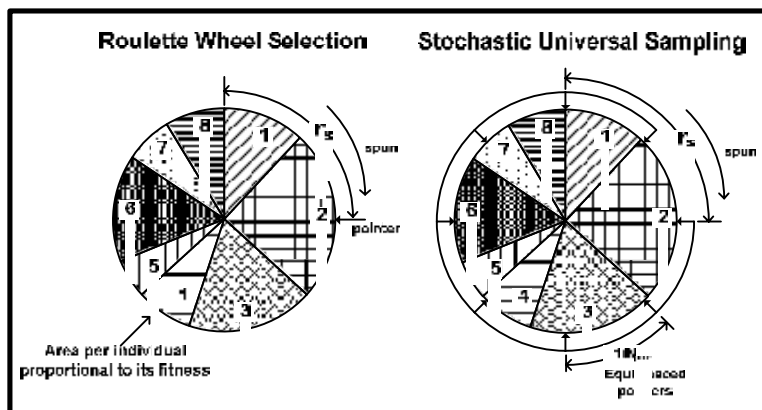


Figure 2.5. Roulette Wheel Selection and SUS

Since individuals with higher fitness occupy a bigger area, they have more chances of being selected multiple times. The segment's size allocated to each individual does not change during the entire selection process. Thus, any individuals could be chosen many times, even to fill a large part of the next generation, which would affect diversity. Several variations of this mechanism are available, one being the Stochastic Sampling with Partial Replacement (SSPR), which reduces an individual's segment every time this is selected (Chipperfield et al., 1994). Another option is the Stochastic Universal Sampling (SUS). This, rather than one single pointer, has N equi-spaced pointers equal to the number of solutions to select (usually the size of the population). The operator is completed in one single run, generating one random number indicating the position of the first pointer is generated, which moves the entire set of pointers, indicating the chosen solutions (Fig. 2.5). SUS is used with the aim of achieving the objectives of reducing stochastic selection errors as much as possible, and to reflect through the assigned fitness of individuals their expected eligibility for reproduction. Reeves and Rowe (2002) indicate that SUS and tournament selection are considered more efficient than RWS.

2.4.3.4. Crossover

Crossover together with mutation constitute the variation operators of the GA. They carry out the exploration process of the GA, creating new individuals that will be later introduced into the population and therefore into the search space. It is in the variation operators where the major differences between genetic operators for binary and real codifications take place.

Crossover is also called *Recombination*. The operator picks two parent solutions from the mating pool and exchanges portions of their chromosome strings (i.e. genetic features) to create two offspring. This is equivalent to using the parent's genes to create new individuals, in the hope that they will retain the best genetic information and be fitter. The simplest recombination is the Single Point crossover (Fig. 2.6), in which a cross point is randomly chosen. The two parents exchange their portions of chromosome indicated by the crossover point. Variations of this operator include the Double Point crossover and the Multiple Point crossover. As its name indicates, the latter consists in randomly choosing multiple cross points for exchanging the portion of the parent's strings. The Uniform Crossover (Fig. 2.6) operates by choosing every bit from the other parent with a probability p (usually $p=0.5$) to create the offspring (Deb et al. book). To implement it, a mask of bits can be randomly created, which decides whether a particular bit is chosen from one parent (0) or the other (1). The mentioned crossover operators are for binary codifications. They can be used for real numbers, but they are quite unsophisticated. Some adaptations of them are used instead.

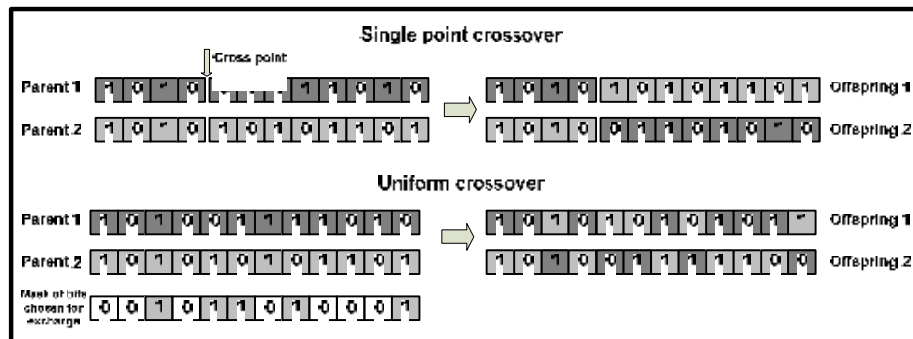


Figure 2.6. Crossover methods for binary numbers

With the objective of preserving some of the good solutions put by the selection process in the mating pool, not all of them are picked for recombination. This is indicated by a crossover probability p_c (typically high, e.g. $p_c \geq 0.6$) that determines which percentage of the mating pool's individuals are to be recombined. A random number R is generated, and if $R < p_c$ no crossover takes place and the children are simple copies of the parents (Marseguerra et al., 2006). This determines a crossover rate, which becomes one more of the parameters of the GA to be tuned.

Methods for real codes are more sophisticated. An account can be found in Haupt & Haupt (2004). They practically enclose the real crossover method into two categories. The first category of methods swaps sections of the chromosomes of the parents, similar to binary methods. One or more points are randomly chosen as cross points and the segments are interchanged. The most elaborate of these methods is the real version of the Uniform Crossover, called Discrete Recombination. This works in the same fashion as the binary version. These methods have the drawback that they do not introduce any new information into the offspring. This is in contrast to binary codes where the phenotype values themselves are altered.

The second types of methods use algebraic formulae that combine the values of the two parent variables into a new variable value. This technique intends to enhance the introduction of new genetic material. These are a group of algorithms generally referred to under the generic name of Blending methods (Haupt & Haupt, 2004). This is a generic name, and some other denominations specific to each algorithm are found. A representative example given by Haupt & Haupt is the function:

$$p_{new} = \beta \cdot p_{mn} + (1 - \beta) p_{fn} \quad (2.8)$$

Where p_{new} is the n^{th} variable of the offspring, β is a random number in the interval $[0, 1]$, and p_{mn} and p_{fn} are the n^{th} variable of the mother and father chromosomes respectively. The complementary value of β (i.e. $\beta-1$) can be used to generate the second offspring. In general, Blending methods use this kind of algebraic functions with some particular variations. Two methods implemented by Mulenbein & Schlierkamp-Voosen (1993) are the Extended Intermediate recombination (EIR) and the Extended Line recombination (ELR), both illustrated in Figure 2.7. The EIR intends to produce new phenotypes around and between the values of the parent's phenotypes:

$$\begin{aligned} p_{new}1_i &= p_{mi} + \alpha_i(p_{fi} - p_{mi}) \\ p_{new}2_i &= p_{fi} + \alpha_i(p_{mi} - p_{fi}) \end{aligned} \quad (2.9)$$

Where $i=1, \dots, n$ represent each variable (gene) of the chromosome, and α is a random number in the range $[-0.25, 1.25]$.

The ELR uses only a single value of α for all variables, creating a point in the line defined by the parents and the perturbation α .

$$\begin{aligned} p_{new}1_i &= p_{mi} + \alpha(p_{fi} - p_{mi}) \\ p_{new}2_i &= p_{fi} + \alpha(p_{mi} - p_{fi}) \end{aligned} \quad (2.10)$$

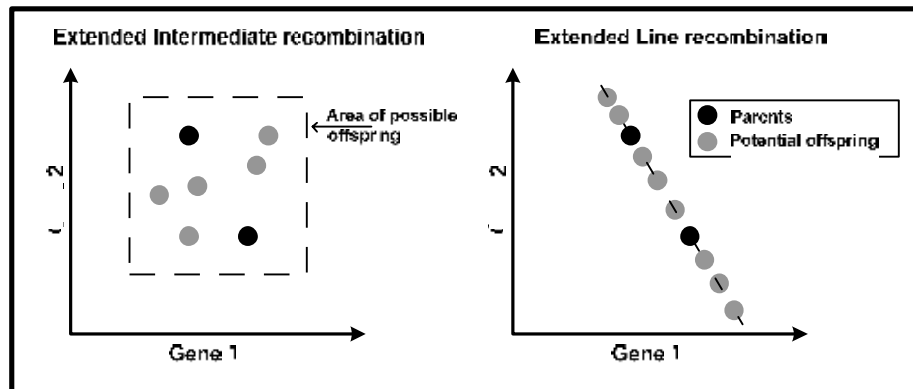


Figure 2.7. Recombination methods for real numbers (Chipperfield et al., 1994)

Geometrically, Discrete Recombination generates a defined hypercube using the parent's values. EIR makes it possible to create offspring inside the volume of the hypercube larger than the one defined by the parents (constrained by α). ELR achieves the same but only along a line. This is shown in Figure 2.7. Other interesting methods are the Blend Crossover (BLX- α) and the Simulated Binary Crossover (SBX), the latter created by Deb (see Deb, 2001).

The adequacy of choosing one blending method over another is quite context-related. Deb (2001) reports that one of his studies found similarities in all these methods. However, it seems that Blending methods are indeed advantageous over the methods that simply swap segments of the parent's chromosomes. Since the selection operator makes multiple copies of some solutions and deletes others, it reduces the diversity of the population. It is clear that Blending methods permit this diversity to be steered more than the other methods, since they introduce variability in the new chromosomes, restoring somewhat the balance between reduction and improvement of diversity.

2.4.3.5. Mutation

Mutation is the second variability operator. This operator randomly changes one of the genes in the new offspring's chromosomes. Typically, this is done with low probability, in the range of 0.001 and 0.01 (Adra, 2007). Reeves & Rowe (2002) recommend using an adaptive mutation rate, although Haupt & Haupt (2004) report no significant difference between varying and constant mutation rate. A mutation rate that is frequently recommended is a fixed rate of $1/l$ (where l is the length of the chromosome). The simplest binary mutation operator is the bit-flipping mutation (Fig. 2.8), where the bits of a chromosome are simply flipped with a certain probability p_m . Mutation is an operator that has responsibility to keep diversity in the population, and it ensures that all the sub-regions in the search space are explored. It contributes

also to escape local minima. The mutation rate must, however, be low since a high mutation rate could convert the search process into a plain random search.

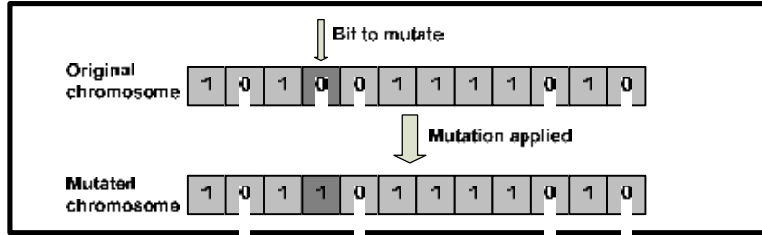


Figure 2.8. Bit-flipping mutation

Mutation operators for real numbers are more sophisticated. Mutation is performed by either perturbation of the gene values or choosing randomly new values within a permitted range (Chipperfield et al., 1994). For mutation of real numbers, higher mutation rates (than for binaries) are advised, between 5 and 20% (Haupt & Haupt, 2004). There are many variations of the mutation operator. For the sake of brevity only one example is discussed here. One method is to add a normally distributed random number (bounded within a specific range) to a variable. There is no specific generic name for these algorithms, so we simply call them “mutation for real numbers”.

A mutation operator proposed by Mulenbein & Schlierkamp-Voosen (1993) for real numbers goes as follows:

$$p_{newi} = p_{previ} \pm range_i \cdot m_{shr} \cdot \delta \quad (2.11)$$

The individual p_{new} is the new mutated solution from the preceding one p_{prev} . The sign + or (-) is chosen with a probability of 0.5. $range$ determines the mutation range established by the variable search interval multiplied by a reduction factor:

$$range_i = R_f \cdot Search_i = R_f \cdot (ul_i - ll_i) \quad (2.12)$$

Where R_f is the reduction factor, $Search_i$ the search interval of the variable and ul and ll the upper and lower limits of the variable. The factor m_{shr} is a mutation-range shrinking factor introduced by Chipperfield et al. (1994), with default value of 1. The factor δ is calculated from a distribution:

$$\delta = \sum_{i=0}^{m-1} \alpha_i 2^{-i} \quad (2.13)$$

The factor α_i is equal to zero before mutation. It acquires a value =1 with probability $p_\delta=1/m$. With this it is expected that in average only one term $\alpha_i=1$ contributes to the sum (i.e. $\delta=2^j$, for $i=j$). This algorithm permits location of the optimal p_{previ} up to a precision of $range_i \cdot m_{shr} \cdot 2^{1-m}$. It generates any point in the hypercube with centre in p_{previ} defined by $p_{previ} \pm range_i$. Mulenbein & Schlierkamp-Voosen (1993) propose values of $R_f=0.1$ and $m=16$ (since they do not include m_{shr} , it would equivalent to 1), while Chipperfield et al. (2004) implement the algorithm with $R_f=0.5$, $m=20$, and user-defined m_{shr} . A deeper analysis of the algorithm is provided in the references given.

2.4.3.6. Reinsertion

Reinsertion is the operator responsible for implementing what is called *selection for survival*. This determines which of the individuals will pass to the next generation at the time that they replace others. Marseguerra et al. (2006) categorize the replacement schemes with three alternatives:

- Fittest individuals. The two parents are replaced by the two fittest individuals involved in the recombination from both the parents and offspring pools. When the selection process discards weak individuals, this mechanism is not recommended because the fittest individuals would propagate constantly to the next generations.
- Weakest individuals. The weakest individuals in the entire population are replaced by the offsprings. This is useful in a large population, since it reduces the presence of weak individuals in subsequent populations.
- Random replacement. Individuals randomly chosen from the previous population are replaced by the offspring. This is quite effective in small populations because it favours a deeper search.

It can be noticed that the first two replacement schemes implement a form of elitist strategy, since they allow the preservation of the fittest individuals in the next generation.

The selection of individuals can be made from both populations, parents and offspring, or only the offspring. This is referred as $(\mu+\lambda)$ and (μ, λ) respectively, where μ represents the parent population and λ the population of offspring. It is a common strategy to produce (or reinsert) fewer offspring than the total number of individuals in the parent population. In this case, the *generation gap* is the fraction of the parent population that is permitted to be reproduced or the fraction of the population that is replaced at each generation (Back et al. 2000). Therefore, a generational gap $G=1$ indicates that the entire population is replaced.

2.4.3.7. Termination criteria

The termination of the iterative process can be done based on three basic criteria:

- The exhaustion of certain predetermined number of generations. In this case, the *number of generations* practically becomes a tuning parameter. This is the most commonly used termination criteria for its simplicity.
- Based on the fitness of the population. It can be that either the mean fitness of the population achieves an established value, or that the fitness of the best individual does. It can also be that the fitness of the population does not have significant evolution.
- Based on the genetic diversity of the population. Here the diversity drops below an assigned limit.

2.4.4. Multi-objective Genetic Algorithms performance questions

2.4.4.1. Exploration vs exploitation

Perhaps the most pressing issue in GA's performance is the right balance exploration versus exploitation *e-e balance* (Purshouse, 2003). Good exploration means to search all possible sub-regions on the search space, while a good exploitation requires making good exploration in the neighbourhood of identified good solutions. The variation operators are considered to provide the exploration capability of the GA, while the exploitative capability resides mainly in the selection operator. According to Deb (2001), premature convergence may be a consequence of excessive exploitation of determined solutions. In contrast, an excessive emphasis on the exploration may considerably increase the solution time, even to exhibit the behaviour of a random search. The clear cut division of exploration and exploitation operators may be quite relative. Consider for example the case of Blending crossover. These methods can implement exploitation and enhance exploration at the same time.

2.4.4.2. Proximity (convergence)

It was mentioned above that an optimizer must have the two basic properties of proximity and diversity. Proximity, also called convergence, is the prime requirement of a GA. Premature convergence to a local optima or having a poor approximation to the real Pareto-optimal front is a considerable issue in GAs. Fitness assignment and selection (for survival) are the main mechanisms for promotion of good proximity. They must steer the search process towards the Pareto-optimal front, thus their importance is fundamental. In addition, the implementation of elitism is another important strategy added to the selection process. The active implementation of elitism is a characteristic feature of the second-generation GAs. Some comparative studies; e.g. Zitzler et al. (2000), have found that the performance of GAs with elitism is notably better than those without it. These features are the main mechanisms for promotion of good proximity.

However, additional measures contribute to the compliance of this objective. Mutation helps to escape local optima, and then premature convergence. An adequate population size and number of generations are necessary to achieve convergence.

2.4.4.3. Diversity

Diversity is the second most important requirement of a GA. The mechanism of fitness sharing for niche formation discussed above is one of the most effective strategies for promotion of diversity. The main difficulty for its implementation is the choice of the niche size σ_{share} . Second generation GAs have proposed alternative mechanisms, such as nearest neighbour density estimation techniques. These include the *clustering* technique of the SPEA (Zitzler & Thiele, 1999), and the *crowding* technique of the NSGA-II (Deb et al., 2002). The crowding technique is based on the computation of a *crowding distance* for each solution. It is notable because it eliminates the need of using a user-defined parameter such as σ_{share} . The crowding distance is the sum of averages of the Euclidean distance of each solution to the two closest solutions at both sides in the same front per objective.

Additional mechanisms for promotion of diversity have been proposed in different GAs. Deb & Goel (2001) argued that diversity must be promoted not only along the Pareto-optimal front, but lateral to it as well, which in turn ensures better convergence. Since a strong elitism may reduce lateral variability, they proposed a lateral-diversity preserving mechanism of *controlled elitism*. This controls the extent of exploitation in benefit of diversity. The crowding technique and the controlled elitism mechanism will be further explained when studying the NSGA-II algorithm in Chapter 5.

The mentioned additions tend to complement the GAs for promotion of diversity. It must not be forgotten, however, that the right choice of fundamental attributes of the GAs is important, such as the mutation rate and the population size. Mutation rate is the standard operator for diversity promotion, which permits the exploration of all regions of the search space. In addition, a too small population can lack of sufficient diversity for the algorithm to explore the entire search space.

2.5. CONCLUDING REMARKS

This chapter has introduced the problem of multi-objective optimization, in which several objectives are simultaneously optimized in order to find the best solutions of a problem. This permits better and more solutions to be presented to the decision maker than when treating the problem as single objective, where one objective was selected as the one with major priority and

the others introduced as constraints. Multi-objective optimization proves a series of best trade-off solutions called the Pareto-optimal set.

Optimization of RAMS+C for safety systems is a complex non-linear integer, combinatorial and stochastic problem. Many times it is difficult to define the dependability functions in explicit analytical form. It is here when the advantages that GAs convey for efficiently treating ill-behaved problems can be exploited. GAs mimic the natural evolution process based on stochastic search techniques. They are able to handle high dimensional, non-linear and discrete problems with discontinuous functions. They produce a pool of several optimal solutions which provides great flexibility for decision-making. Two of the most efficient multi-objectives GAs are the Fonseca & Fleming MOGA and the NSGA-II. The advantages of these two GAs is exploited in this thesis for solution of SIS optimization problems. They are used for implementing the optimization cases of chapter 3 to 6, where a detailed description of the working principles of these algorithms is provided.

RAMS+C multi-objective optimization with GAs has a recent history, where several aspects of it has been approached in the search for better designs and test and maintenance polices. An overview of these approaches has been provided in this chapter, and some issues not fully explored have been detected. The optimization cases previously addressed by other researchers have not comprised the compliance with the requirements with the international standards IEC 61508, including the level of modelling detail necessary for real-life SIS. Detected niches and opportunities for research of system optimization with GAs that will be explored in this work include modelling of Common Cause Failure and diagnostic coverage, integration of diverse redundancy for improvement of system performance, time-dependent modelling for more precise modelling of unavailability in optimization of test intervals and strategies and integration of Moon voting architectures optimization of system design and test.

CHAPTER 3

Optimization of SIS design with parallel redundancy

This chapter presents an initial optimization case by Genetic Algorithms of a SIS design based on safety and reliability measures plus Lifecycle Cost. It combines the theory of safety systems, the standard IEC 61508, dependability modelling and GAs presented in the previous chapters and demonstrates how the overall methodology works. The standard IEC 61508 establishes the requirement for safety-related systems to meet specific Safety Integrity Levels (SIL). The SIL is determined in terms of Average Probability of Failure on Demand (PFD_{avg}) for systems that operate in low-demand mode. This optimization takes into account the level of modelling detail contemplated by the standard, including multiple failure modes, diagnostic coverage and Common Cause Failures. This chapter addresses the case of series-parallel systems. Optimization is approached by treating the problem as one of redundancy allocation and component selection together with some test specifications. Modelling is made using Fault Tree Analysis with house events. The Multi-Objective Genetic Algorithm proposed by Fonseca & Fleming (1993) is used as optimization technique.

3.1. OPTIMIZATION OF SAFETY SYSTEM'S SPECIFICATIONS

An overview of optimization of safety systems with GAs was provided in the previous chapter. Therein, it was established that multi-objective optimization of SIS for process industry has not been thoroughly investigated yet. First of all, no optimization compliant with IEC 61508 requirements have been addressed so far. This also means that the required level of modelling detail for dependability quantification of safety systems within the frame of the standard has been consistently omitted. Common Cause Failure is usually ignored, and distinction of dangerous and safe failure modes has only been taken into account by a few studies. Diagnostic coverage, a fundamental feature of components used in safety systems, has never been mentioned in any optimization study. Some studies have included safety-related costs or risk-based measures (such as exposure times or frequency of hazards) in the optimization (e.g. Yang et al., 1999; Cantoni et al., 2000; Giuggioli-Busacca et al., 2001). However, these do not treat directly the specific measures of SIS dependability, like quantifying system safety with the average Probability of Failure on Demand, and even more rarely considering the Spurious Trip Rate. The only studies that have considered measures of dangerous and safe failures together for optimization against cost are Pattison & Andrews (1999), Andrews & Bartlett (2003), and more recently in multi-objective optimization Borisevic & Batlett (2007a, b) and Riauke & Bartlett

(2008). Their study cases (the High Integrity Protection System and the Water Deluge System) are maybe the closest applications to the field of SIS for process industry. However, they did not approach either the IEC 61508 requirements or the necessary full detail in the modelling. In addition, several multi-objective GAs have been applied to the solution of reliability problems. However, the GA developed by Fonseca & Fleming (1993) has not been explicitly found in any related application. This opportunity for research is explored here.

The problem explored in this chapter is the optimization of safety system design with parallel architectures. It includes redundancy allocation, component selection (discrete reliability allocation) and Test Intervals. It addresses the requirements of IEC 61508. Thus, it contemplates the level of modelling detail necessary for quantification of dependability: type of hardware, multiple failure modes, diagnostic coverage and Common Cause Failure (CCF). This initial optimization case includes optimization of dependability with three objectives: Probability of Failure on Demand (PFD_{avg}), Spurious Trip Rate (STR) and overall system (maximum) unavailability. This addresses the negative effects of both dangerous and safe failures. The fourth objective is Lifecycle Cost, which full model's first formulation is made herein.

3.2. FAULT TREES WITH HOUSE EVENTS

Fault Tree Analysis is the method used for modelling dependability measures. Its advantages over other methods were detailed in the previous chapter, such as its versatility for modelling large complex systems, its functionality for documenting and visualizing the failure mechanisms and its relative construction simplicity. Based on these advantages it has been chosen as the dependability quantification the method. Optimization of system design requires the evaluation of many potential designs in order to find the optimum. This would require the construction of one fault tree for every single solution, which would be complicated and time-consuming. It is, therefore, necessary to empower the fault tree with the capacity to accommodate changes in the design, so that one single tree can be used for modelling all potential solutions. This has been achieved using house events.

House events (see Fig. A.2) are logic external events that take only a value of true or false (1 or 0). They are not completely new to Fault Tree Analysis, and were already included by Vesely et al. (1981). However, Andrews (1993) suggested their application for turning sections of fault trees on and off so that changes in the design of safety systems could be accommodated. This was later applied with GAs in subsequent works of his research team (Pattison & Andrews, 1999; Andrews & Bartlett, 2003; Borisevic & Bartlett, 2007a, b; Riauke & Bartlett, 2008). The logic behind house events can be better understood observing Figure 3.1. This figure shows the

modelling of failure of a system that has one component A, and the possibility of having one or two more components, B and C. If the house events of B or C are activate (=1), these components are fitted into the system's tree and then their failure contributes to the system's failure. Also notice that the CCF event is activated by a house event that indicates that the redundancy is higher than 1. Fault trees with house events are used for modelling safety and reliability in the following sections.

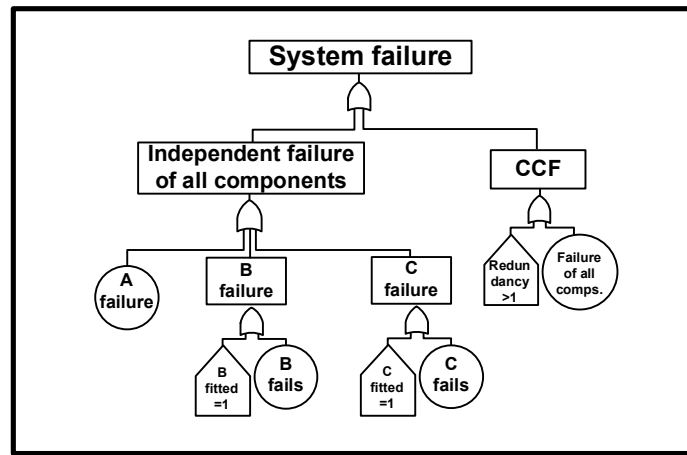


Figure 3.1. Example of application of house events

3.3. THE LIFECYCLE COST MODEL

The model presented here is largely based on Goble (1998), and it has been modified and adapted for the necessities of this chapter. The overall Lifecycle Cost, given by Eq. (3.1), is split into three main factors: procurement, operation and risk costs. The annually adjusted financial cost is considered by calculating the operating and risk costs in annuities at present value (Eq. (3.2)) for the operating life of the system. The present value of a future expense is an equivalent estimation of how much money is necessary to invest now in order to pay for the expense in the future.

$$LCC = C_{PROC} + (C_{OP} + C_{RISK}) \cdot PVF \quad (3.1)$$

Where the applied factor by present value is:

$$PVF = \frac{1 - (1 + R)^{-T}}{R} \quad (3.2)$$

The term R represents the discount rate, which combines both the interest rates and inflation rates (e.g. high-risk projects are given higher discount rates). T is the useful life of the system (mission time) in years.

Procurement cost per year is quantified as:

$$C_{PROC} = \sum_{\forall i} (C_i^{purchase} \cdot \beta_{CM-P} + C_i^{design} \cdot \beta_{CM-D} + C_i^{inst/comm} \cdot \beta_{CM-I}) N_i + C_{Start-up} \quad (3.3)$$

Where N_i is the number of components of the i^{th} subsystem. Basically, these are the decision variables of the problem, and:

C^{design} = Design cost

$C^{purchase}$ = Purchase cost

$C^{inst/comm}$ = Installation & Commissioning cost

$C_{Start-up}$ = Initial plant start-up

The terms β_{CM} correspond to β factor cost modifiers (where P, D, and I stand for purchase, design and installation respectively). These are explained in Section 3.5.

The overall operating cost per year (Eq. (3.4)) includes consumption (C_{cons}), preventive maintenance (C_{PM}), testing (C_T) and corrective maintenance (C_{CM}). These three latter factors are respectively calculated in Eqs. ((3.5)-(3.7)).

$$C_{OP} = C_{cons} + C_{PM} + C_T \quad (3.4)$$

Where:

$$C_{cons} = \sum_{\forall i} (C_i^{cons} \cdot \beta_{CM-C}) N_i \quad (3.5)$$

$$C_{PM} = \sum_{\forall i} C_i^{PM} \cdot N_i \quad (3.6)$$

$$C_T = \sum_{\forall i} \frac{1}{TI_{ij}} C_i^T \cdot N_i \quad (3.7)$$

Where C_i^{cons} is the consumption cost per year, C_i^{PM} is the cost of preventive maintenance per year, and C_i^T the cost of test per event, all of them per component. β_{CM-C} the consumption cost modifier, TI is the test interval per year.

The risk cost per year comprises both the cost per spurious trip rates and the cost of hazards (Eq. (3.8)). This cost must be certainly considered. It could be necessary to be paid in the form of insurance premiums.

$$C_{RISK} = C_{STR} + C_{HAZARD} \quad (3.8)$$

The cost caused by safe failures (STR) is proportional to the cost of production loss from each spurious shutdown and the cost of repair of those failures:

$$C_{STR} = [(\sum_{\forall i} C_i^{CM} + SD_{Loss}) SD_{Time} + \sum_{\forall i} C_i^{spares} N_i] \cdot STR \quad (3.9)$$

Where:

$$C_i^{spares} = \%PCE \cdot C_i^{purchase}$$

C_1^{CM} = Cost of repair per hour

SD_{Loss} = Loss of production per hour

SD_{Time} = Plant restoration downtime after spurious trip

C_1^{spares} = Cost of spare consumption per event

%PCE = Fraction of component purchase cost

The cost of hazard includes the cost of a catastrophic accident modified by the frequency of estimated plant failure frequency and the SIS $PF_{D_{avg}}$:

$$C_{HAZARD} = C_{ACC} \cdot F(ACC|PF_{D_{avg}}) \cdot PF_{D_{avg}} \quad (3.10)$$

Notice that the term $F(ACC|PF_{D_{avg}})$ is the frequency of plant failure (or accident) per year without the SIS, which is actually the frequency of plant failure assuming the SIS is unavailable: i.e., by the Total Probability Theorem, it is the $F(ACC|PF_{D_{avg}})$ term in the expression:

$$F(ACC) = F(ACC|PF_{D_{avg}}) \cdot PF_{D_{avg}} + F(ACC|1 - PF_{D_{avg}}) \cdot (1 - PF_{D_{avg}}) \quad (3.11)$$

The cost of an accident C_{ACC} can be determined using the sum of different factors. This thesis considers that the fundamental factors to be considered are: plant assets loss, deferred production and liability (i.e. fatalities) costs. Including the cost of liability contemplating the potential loss of life is usually a controversial affair, but to include it is useful to reflect the benefit of preventing fatalities. The Health and Safety Executive (HSE, 2001) approaches this by using the Value of Preventing a Fatality (VPF). The VPF is problem-specific. The HSE gives as example the figure of £1,000,000 (2001 figure) used in the appraisal of road safety measures. A reader interested in estimating the C_{ACC} is encouraged to consult this reference.

Figure 3.2 summarizes the entire LCC model.

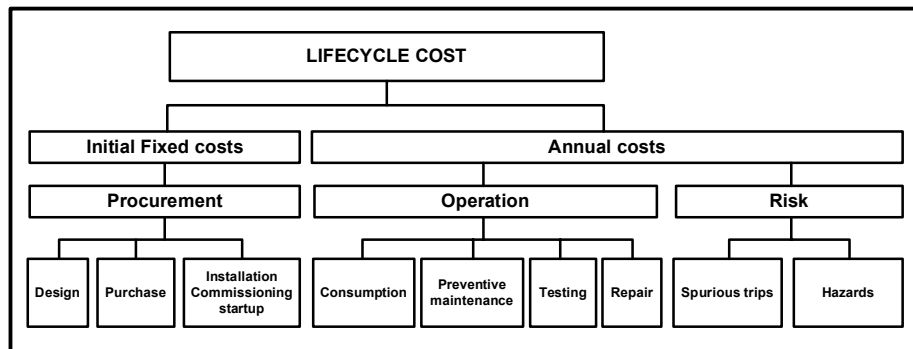


Figure 3.2. Safety System Lifecycle Cost

3.4. FONSECA & FLEMING MULTI-OBJECTIVE GENETIC ALGORITHM

The Fonseca and Fleming Multi-Objective Genetic Algorithm was formulated in Fonseca & Fleming (1993). Some additional developments were proposed in Fonseca & Fleming (1995a, 1995b, 1998). This algorithm was empowered for multi-objective optimization and named Multi-Objective Genetic Algorithm (MOGA). As mentioned in Section 2.4, this is accepted as being the most efficient algorithm of the first generation. The MATLAB[®] toolbox used for the present research work is documented in Chipperfield et al. (1994). MOGA was one of the first genetic algorithms to be proposed to approach multi-objective problems based on the concept of Pareto dominance.

The Fonseca & Fleming MOGA has the following specific characteristics:

- Ranking based on Pareto dominance
- Fitness assignment based on dominance-rank ranking
- Option to implement niching by fitness sharing, and mating restriction
- Selection by stochastic universal sampling
- Suggested a progressive preference-based method, incorporating the Decision Maker with the GA
- Uses parallel coordinates for visualization of the trade-off surface in two dimensions

3.4.1. Pareto-based ranking.

Since a cost is calculated for each member of the population, ranking is made after the population is sorted based on the cost (a measure of utility) of each individual. The Fonseca & Fleming (1993) proposition is to assign an individual's rank equal to the number of individuals by which it is dominated. Therefore, the non-dominated solutions have all the same rank. This ranking technique allows adaptation of the algorithm for search of the Pareto front in both convex and non-convex trade-off surfaces. The original approach was to compute the rank:

$$\text{rank}(x_i, t) = p_i^{(t)} + 1 \quad (3.12)$$

Where x_i is an individual at generation t dominated by p_i individuals. This ranking method is illustrated in Figure 3.3. It makes the non-dominated individuals to be ranked one (although it is frequently applied without summing 1, so that the non-dominated individuals are ranked zero). A second proposition for ranking was made later (Fonseca & Fleming 1995a, 1998), see Eq. (B.6) in Appendix B.

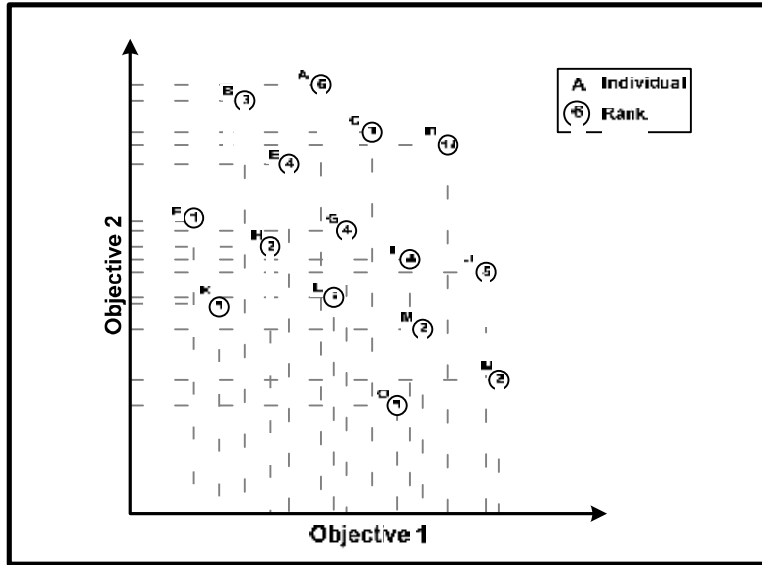


Figure 3.3. Pareto-based ranking in MOGA

3.4.2. Fitness allocation

As mentioned in Chapter 2, fitness assignment in the MOGA is based on dominance rank. The concept of *average fitness* is used by Fonseca (1995) to describe *the fitness of an individual normalized by the average fitness of the population*. This gives a good direct indication of how much better, or worse, than the current average individuals are.

The MOGA uses rank-based fitness assignment, where the fitness is mapped against the rank values in a monotonic fashion, and where the fitness must be non-negative values.

The method proposed in Fonseca & Fleming (1993, 1995a, 1998) is:

1. The population is sorted by rank order (i.e. cost).
2. Fitness is allocated by a mapping with the rank (from best to worst ranked individual); i.e. position in the population, according to a pre-defined function. The mapping may be linear or exponential. Fitness must be non-negative in all cases. This makes the fitness assignment independent from the scale of the problem, so performing scaling is not necessary.

From Fonseca (1995), the function for the linear assignment could be:

$$f(r) = s - (s - 1) \cdot \frac{2r}{N_{pop} - 1} \quad (3.13)$$

Where s ($1 < s \leq 2$) is the *relative fitness desired for the best individual*. The sum of all fitness values must be equal to the population size: $\sum_{i=0}^{N_{pop}-1} f(i) = N_{pop}$

The exponential assignment formula proposed by Fonseca (1995) is:

$$f(r) = \rho^r \cdot s \quad (3.14)$$

Where $s > 1$ and ρ is determined so that $\sum_{i=0}^{N_{pop}-1} \rho_i = N_{pop} / s$. Exponential mapping is more flexible than linear mapping since there is no upper bound for s .

According to Fonseca (1995), the exponential assignment contributes to a more diverse search because, different from the linear mapping, it does not penalize the worst individuals too much and it assigns middle individuals fitness values slightly less than average.

3. The fitness of individuals with the same rank is then averaged. This permits all individuals with the same rank to be sampled at the same rate while keeping the global population fitness constant.

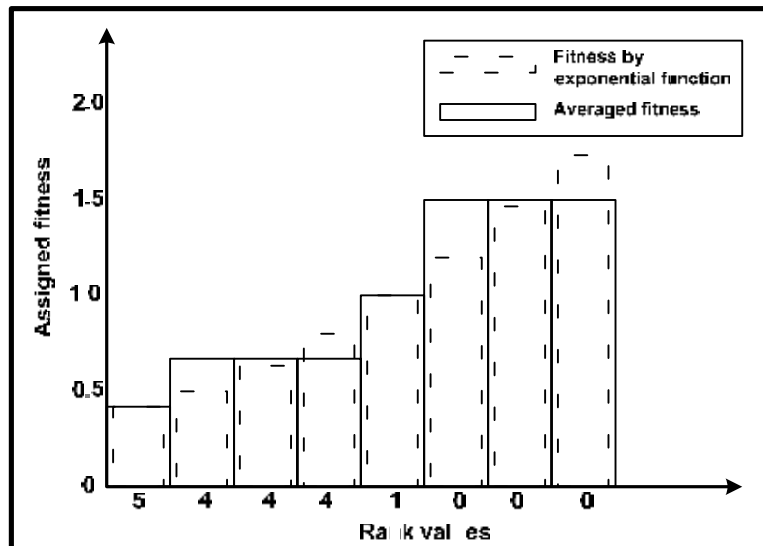


Figure 3.4. Fitness assignment by exponential function and average fitness (Fonseca, 1995)

Figure 3.4 (reproduced from Fonseca, 1995) illustrates an example of a population of 10 individuals, where they are sorted by they rank. The fitness values assigned with an exponential function are represented by the narrower bars. The wider bars represent the averaged fitness of the individuals with the same rank.

Ranking-based fitness assignment allows the best individual in the population to be invariably given the same relative fitness ($\leq s$). Thus the excessive reproduction of potential “*super*” individuals is averted. In the same fashion, the best individual is still consistently preferred from the others when all of them perform similarly well.

3.4.3. Presentation of results

Fonseca & Fleming (1993, 1998) make use of the parallel coordinates as the graphical representation of the trade-off data. This is similar to the Value Path Method (see Deb, 2001). The method of parallel coordinates is the representation of each non-dominated solution from the Pareto-optimal set in two coordinates, being each point in the X-axis one of the objectives, plotted against the normalized objectives in the Y-axis. Each point is united to one another by a line (each line representing one solution), where lines run non-concurrently they represent non-competing objectives, while crossing lines make evident conflicting objectives. An example is shown in the Discussion of Results section, Figure 3.12.

3.5. DESCRIPTION OF THE APPLICATION PROBLEM

The problem at hand is the design of a Safety Instrumented System for a process plant. The safety system, illustrated in Figure 3.5, is for a hypothetical compression system of a LNG distribution plant. A compressor’s outlet pipeline must be protected against leakages. The detection system is comprised by a pressure measurer/transmitter subsystem (PT), a logic solver subsystem (LS) implemented by a PLC, and a shut-down valve as Final Control element subsystem (FC). If the LS detects a considerable fall in pressure (measured by the PT), the FC valve must be shut immediately to interrupt the flow of gas. A failure to interrupt a leakage could have as a consequence a vapour cloud formation within the confined conditions of the compressor house, which in case of being ignited would have catastrophic consequences (a vapour could explosion). It is assumed that the compressor itself has its own protective system, and the safety function approached is the last line of defence to prevent the leakage. Table 3.1 shows all the relevant data. Notice that failure rates (λ) are split into four different modes (see the first column).

A variety of sources have been consulted to put together a realistic case study based on similar equipment. This included books (Goble, 1998; Smith, 2005; CCPS, 2000), standards (ISA, 1999), databases (Hauge et al., 2006b), test reports or real equipment data publicly available (Medoff, 2007), and engineering judgement (especially for estimating some costs). The same has been done for the case studies presented in following chapters.

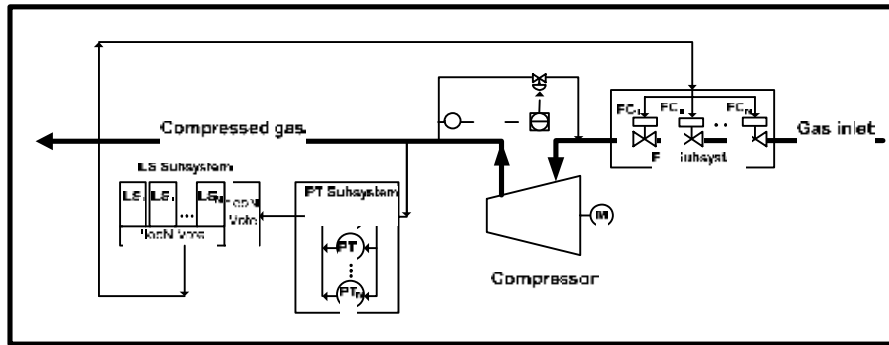


Figure 3.5. Leakage protection system of a natural gas compressor

The figure for the Catastrophic loss cost presented in Table 3.1 has been estimated for a hypothetical case, summing up the cost for loss of the plant's facilities of 3.6×10^6 , the cost of loss of production for the entire plant's useful life of 240×10^6 , and the estimated total value of preventing fatalities of 50×10^6 ($VPF = 1 \times 10^6 \times 50$), all of them in monetary units. Notice that the number of potential fatalities must always be estimated based on a rigorous hazard analysis, consequence modelling and quantitative risk assessment.

The criticality of the system being analyzed requires high integrity. This can be enhanced by changing components, increasing redundancy (as indicated in the figure), or changing the design for reduction of CCF. In this thesis CCF is quantified using the β factor model. The model and the phenomenon of CCF are discussed in detail in Chapter 4.

The decision variables of the problem are as follows:

1. Redundancy scheme: four options of parallel redundancy: 1, 2, 3 or 4.
2. Type options: There are four optional different types per subsystem (referred to here as Optional-type so it is not confused with the component type classification of IEC 61508, see Section 1.5). Each choice has different diagnostic coverage, failure rate and cost specifications.
3. β factor: This factor indicates the percentage of the total failure rate that will be attributed to Common Cause Failure. Since enhancing the design minimizes the likelihood of Common Cause Failure, the β factor diminishes. For field instruments there are two design options, the baseline case ($\beta = 0.035$) and design with additional electrical separation ($\beta = 0.02$). Field instruments are usually physically separated, so this contributes a baseline case with lower β factor than for the controllers. For the controller there are three options: baseline case (the controllers are located in the same rack and supported by the same electric supply; $\beta = 0.05$),

design with electrical separation ($\beta=0.035$) and design with electrical plus physical separation ($\beta=0.02$).

4. Test Interval: This may be between 1 and 24 months, in steps of one month.

Table 3.1. Dependability and lifecycle cost data

Subsystem	TRANSMITTER				CONTROLLER				VALVE			
	Supplier 1		Supplier 2		Supplier 1		Supplier 2		Supplier 1		Supplier 2	
	1	2	3	4	1	2	3	4	1	2	3	4
Type					standard-PLC safety-PLC standard-PLC safety-PLC							
Diagnostic Coverage (%)	0	75	30	80	57.5(S) / 48.8(D)	99(S) / 98.2(D)	45(S) / 60(D)	90(S) / 95(D)	0	30	25	50
Failure rates data ($\times 10^{-6}/\text{hr}$)												
Safe detected	0	4.3	13.68	36.48	3.894	9.996	1.773	3.852	0	7.2	9.5	19.0
Safe undetected	5.7	1.4	31.92	9.12	2.878	0.098	2.167	0.428	24.0	16.8	28.5	19.0
Dangerous detected	0	17.1	6.84	18.24	2.906	5.956	2.886	4.959	0	5.7	7.125	14.25
Dangerous undetected	22.8	5.7	15.96	4.56	3.048	0.112	1.924	0.261	19.0	13.3	21.375	14.25
Total	28.5	28.5	68.4	68.4	12.726	16.162	8.75	9.50	43.0	43.0	66.5	66.5
Life cycle cost data												
Purchase	35000	50000	25000	32000	60000	90000	75000	100000	25000	30000	25000	30000
Design	800	800	800	800	11000	11000	11000	11000	800	800	800	800
Installation	340	340	340	340	500	500	500	500	400	400	400	400
Consumption per year	140	150	160	160	200	210	180	180	120	120	125	125
Maintenance per year	6000	6000	7000	7000	5000	5000	4000	4000	5000	4000	3500	3500
Repair cost												
(Cost per hour + %purchase cost/event)	50/hr +10%	60/hr +10%	50/hr +10%	60/hr +10%	60/hr +15%	60/hr +15%	50/hr +15%	50/hr +15%	45/hr +7%	45/hr +7%	50/hr +7%	50/hr +7%
Test cost per event	100	70	85	65	250	150	270	200	120	100	105	80
Beta Factor												
Base case	$\beta = 0.035$				$\beta = 0.05$				$\beta = 0.035$			
With electrical separation	$\beta = 0.02$				$\beta = 0.035$				$\beta = 0.02$			
With electrical & physical separation	-				$\beta = 0.02$				-			
Other data												
Repair time = 8 hours												
$SD_{\text{time}} = 24$ hours												
Start up cost = 1000 units												
Shutdown loss cost = 3200 units/hr production loss												
Plant's risk (without SIS) = 5.88×10^{-3} dangerous events per year												
Catastrophic loss cost = 300×10^6 units/event including liability												
Discount rate = 5%												
Mission time T = 15 years												

The overall problem is of minimization of four objectives: Average Probability of Failure on Demand (PFD_{avg}), Spurious Trip Rate (STR), Unavailability and Lifecycle cost (LCC). The problem has approximately 6.8×10^8 potential solutions. This large number of solutions means that evaluation of every single combination in order to find the optimal is not a realistic option. Thus, application of Genetic Algorithms makes a practical approach.

The system is required to achieve a SIL 3, with a goal of maximum PFD_{avg} of 1.7×10^{-4} . The actual risk presented by the plant has been estimated to be 5.88×10^{-3} dangerous events per year, which is the demand rate of the safety system ($F(\text{ACC}|\text{PFD}_{\text{avg}})$). On the other hand, the tolerable risk frequency (F_{TR}) or risk target has been set to 1×10^{-6} per year. The necessary risk reduction is equivalent to the PFD_{avg} , determined by the expression:

$$PFD_{avg} \leq \frac{F_{TR}}{F(ACC|PFD_{avg})} \quad (3.15)$$

According to Table 1.1, a $PFD_{avg}=1.7 \times 10^{-4}$ truly corresponds to a SIL 3. For a detailed account of the quantitative method for determination of the target SIL (which is out of the scope of this work) consult IEC 61508 Part 5.

It is important to notice that Table 3.1 provides a separate diagnostic coverage (ε) for each failure mode for controllers. Eqs. (3.16) and (3.17) give the ε for dangerous and safe failure modes respectively. Notice that in the table the values for controller's diagnostic coverage of dangerous failures ε^D are indicated with a (D), and for safe failures ε^S with a (S). In contrast, transmitters and valves used in this example have the same diagnostic coverage for both failure modes ($\varepsilon^D=\varepsilon^S$), and thus only one single value is indicated.

$$\varepsilon^D = \frac{\lambda^{DD}}{\lambda^{DD} + \lambda^{DU}} \quad (3.16)$$

$$\varepsilon^S = \frac{\lambda^{SD}}{\lambda^{SD} + \lambda^{SU}} \quad (3.17)$$

As it can be seen in Table 3.1, the lifecycle cost data includes both procurement and operation related costs. In addition, consider that the improvement of defences against CCF, which modifies the β factor, affects some costs. Table 3.2 shows the applicable cost modifiers (β_{CM}), indicating the percentage of increment for the affected costs to be applied in Eqs. (3.3) and (3.5).

Table 3.2. Cost increments by design for β factor improvement

Cost modifier per concept	Cost increment	
	Electrical separation	Electrical & physical separation
Purchase, β_{CM-P}	15 %	35 %
Design, β_{CM-D}	5 %	10 %
Installation, β_{CM-I}	10 %	25 %
Consumption, β_{CM-C}	30 %	35 %

Additional data and assumptions necessary for the quantification are given next: all the instruments are considered Type B (see Table 1.2); the system is considered to be in low-demand mode of operation; component failure rates are constant; failure of each subsystem is independent from others; once a component has failed, it remains in that state until it is repaired;

testing and repair are assumed to be perfect (return to normal state and “as new” condition respectively).

3.6. MODELLING AND QUANTIFICATION

Two fault trees were developed to quantify both the PFD_{avg} and the STR, shown in Figures 3.6 and 3.7 respectively. In order to enable the fault trees to flexibly accommodate the changing redundancy design, house events were put in place. It is important to highlight that Figures 3.6 and 3.7 do not show the full fault trees. Only the branches corresponding to the PT subsystem are fully shown. Since the other two branches are identical they are omitted for the sake of brevity. Both fault trees have been solved gate-by-gate using simplified approximation equations in the basic events. This method is suitable since there are no repeated basic events in the trees, thus reducing the complexity and computational effort needed to solve them by other methods. Since the product λt is very small (i.e. $\lambda t \ll 0.1$), the rare event approximation $1 - e^{-\lambda t} \approx \lambda t$ is conveniently used in the equations that solve the fault trees. This lessens the complexity of the gate-by-gate solution implementation and permits a better understanding of the models.

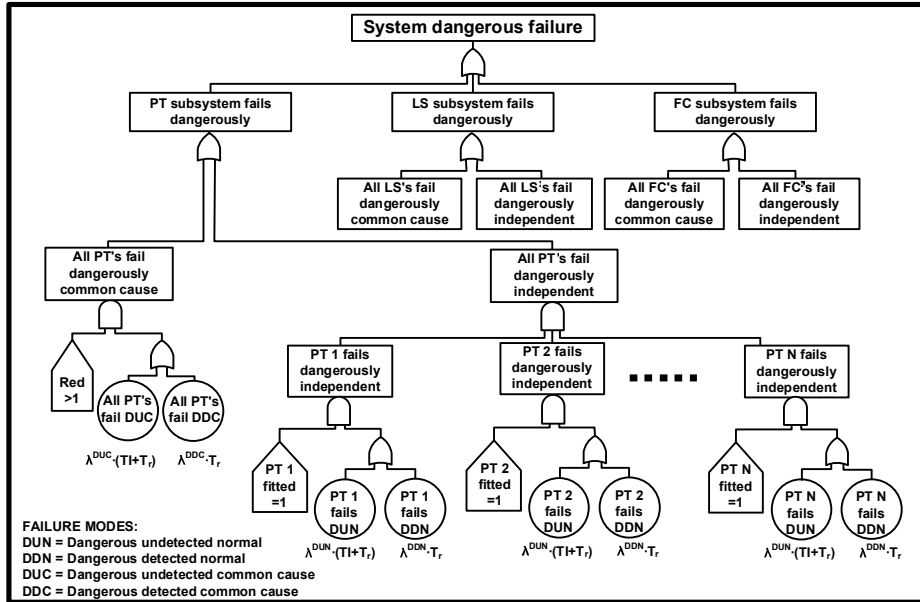


Figure 3.6. Fault tree for quantification of Probability of Failure on Demand

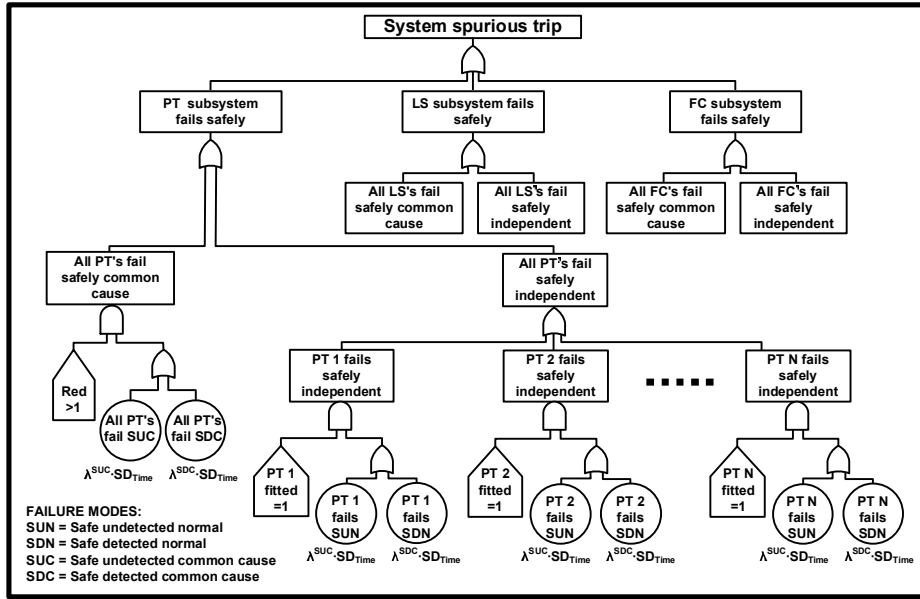


Figure 3.7. Fault Tree for quantification of Spurious Trip Rate

It is important to fully understand the logic in the fault trees. The dangerous failure of any of the three subsystems can develop into the top event of the tree in Figure 3.6. The OR gate that leads to PT subsystem failure indicates that this subsystem can be failed by either the simultaneous independent (normal) failure of all the redundant components or a common cause failure. Notice that each basic event represents a probability of failure (given by a product λt), there being one basic event per each component failure mode (common cause and normal failure, detected or undetected). The fault tree in Figure 3.6 is solved to obtain the PFD. The fault tree of Figure 3.7 can be understood following the same analysis.

The average PFD is obtained integrating the PFD equation with respect to the Test Interval (see Eq. (1.22)). Eqs. (3.18) and (3.19) give the PFD_{avg} for simplex subsystems and for redundant cases respectively.

$$PFD_{avg} = \sum_{\forall i} \left(\lambda_i^{DU} \cdot \left(\frac{TI}{2} + T_r \right) + \lambda_i^{DD} \cdot T_r \right) \tag{3.18}$$

$$PFD_{avg} = \sum_{\forall i} \left(\frac{[\lambda_i^{DUN} (TI + T_r) + \lambda_i^{DDN} \cdot T_r]^{N_i+1} - [(\lambda_i^{DUN} + \lambda_i^{DDN}) T_r]^{N_i+1}}{\lambda_i^{DUN} (N_i + 1) \cdot TI} + \lambda_i^{DUC} \left(\frac{TI}{2} + T_r \right) + \lambda_i^{DDC} \cdot T_r \right) \tag{3.19}$$

The unavailability by safe failure is given in Eq. (3.20), which was derived from the fault tree of Figure 3.2. The STR per year was also derived from Figure 3.2 considering the basic event as only failure rates. This is given by Eq. (3.21), which only shows the terms for the PT subsystem

(for the other two subsystems just add the same corresponding terms). Notice that for redundancy =1 both formulae are the same, since $\lambda^{SUC} + \lambda^{SDC} = 0$.

$$U_{SF} = \sum_{\forall i} (\lambda_i^{SUN} + \lambda_i^{SDN}) \cdot SD_{Time} \cdot N_i + (\lambda_i^{SUC} + \lambda_i^{SDC}) \cdot SD_{Time} \quad (3.20)$$

$$STR = \sum_{\forall i} (\lambda_i^{SUN} + \lambda_i^{SDN}) \cdot N_i + (\lambda_i^{SUC} + \lambda_i^{SDC}) \quad (3.21)$$

Finally, the unavailability is obtained adding the PFD and the PFS up:

$$Unavailability = PFD + U_{SF} \quad (3.22)$$

3.7. IMPLEMENTATION OF THE OPTIMIZATION ALGORITHM

The safety system design was optimized using the Fonseca & Fleming MOGA (2003). Since the decision variables are integer, a binary string was created; which has a total length of 31 bits (Fig. 3.8).

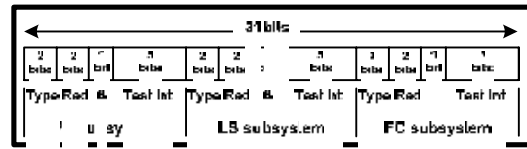


Figure 3.8. Chromosome code for the safety system design

A baseline program was created in MATLAB[®] to use the MOGA toolbox (Chipperfield et al., 1994). The algorithm follows the flowchart of a standard GA (Fig. 1.3). Figure 3.9 shows the pseudo-code of the baseline program. The first step is to create a random population. Then the algorithm is evaluated iteratively until the number of generations is exhausted. This requires decoding the chromosomes into phenotype values, followed by evaluation of the four objective functions. A table of the objectives with the function value vectors of the population is created (subsequently named TOF), to which the GA operators are applied. The parameters used for the program are: Population: 50 individuals; Generations: 300; Generational gap: 0.8 (80% of the population is renewed per generation); Crossover: Single point crossover at 0.7; bit-flipping mutation at 0.1; reinsertion rate 1.0; fitness mapping: exponential (Eq. (3.14)); selection by Stochastic Universal Sampling. An external archive gathers the best ranked individuals per iteration (Table_nondominated). After the maximum number of generations is exhausted, its content is re-ranked to select only the global non-dominated individuals. This gives the Pareto-optimal set of solutions. These results are plotted in graphs, and a table is presented with the solutions of the Pareto-optimal front. Finally, the parallel coordinates plot is generated.

```

Chromosome = Create(Random population)
While Gen < MaxGen {
  Phenotype = decode(Chromosome)
  Objective function {
    PFDavg(Phenotype)
    STR(Phenotype)
    Unav(Phenotype)
    LCC(Phenotype)
  }
  TableObjectiveFunction = [PFDavg STR Unav LCC]
  Genetic Operators {
    Ranked_TOF = rank(TableObjectiveFunction)
    Fitness_TOF = fitness(Ranked_TOF)
    Select_TOF = selection(Chromosome, Fitness_TOF)
    Recombin_TOF = Crossover(algorithm, Select_TOF)
    Mutate_TOF = mutation(Recomb TOF)
    Chromosome = reinsertion(Chromosome, Mutate_TOF)
  }
  Table_nondominated = Table_Nondominated | Table_Nondominated |Chromosome,
  TableObjectiveFunction, Ranked_TOF=0, Gen]
  Gen = Gen+1
}
Ranked_optimal_solutions = Rank(Table_nondominated)
Optimal_solutions = Ranked_optimal_solutions=0
Plot_3D(Table_nondominated)
Plot_2D(Table_nondominated)
Display(Table_nondominated)
Plot_Parallel_coordinated(Table_nondominated)

```

Figure 3.9. Pseudo-code for the Baseline Program

Three different program modalities implementing different approaches, based on the baseline program, have been designed. The difference among the three modalities lies in the way the initial population of the second and subsequent runs of the algorithm is created. The intention is to explore whether creating the initial population by feeding good solutions from the previous run (rather than just randomly) can help to improve the obtained Pareto-optimal set (i.e. more solutions and better distributed), and whether this can also guide the search towards a specific goal. The three modalities are as follows:

- **Modality 1.** The baseline program is run 10 times; every run is independent from one another. Every run the optimal individuals are fed into an external archive. This is finally re-ranked to obtain the non-dominated individuals from all the 10 runs and plotted.
- **Modality 2.** The baseline program is run 10 times, but each new run is fed with the optimal individuals from the previous run as the new initial population. In this way, every run intends to optimize an optimal population from the last run.
- **Modality 3.** The baseline program is run 10 times. After every run, only individuals with SIL 3 or higher are chosen for the new population, which is fed to the next run as the initial population. Every three runs, this population is purged from non-optimal individuals (rank>0).

3.8. DISCUSSION OF RESULTS

Figure 3.10 shows the graphical results of one single run of the Modality 1 program. Notice the difference of scale (linear and logarithmic) between the two graphs for the PFD_{avg} , which is to improve visualization. The graphs show all solutions found in the entire objective space plus the Pareto-optimal set (indicated with black stars). Observe that the PFD_{avg} in the range $>1 \times 10^{-2}$ is not in conflict with the lifecycle cost: the lower the former, the lower the later, which suggests that, in general, introducing the safety system results in a lower LCC since the gross cost of risk is lowered (which is about $\$18 \times 10^6$ without the safety system). However, it is observable that these two measures become conflictive objectives in the optimal front. This means that after a certain point, to reach a lower PFD_{avg} (which could be for the sake of meeting a specific SIL requirement) actually requires an increment of the LCC. This increment in LCC is very sharp after $PFD_{avg} < 1.2 \times 10^{-4}$. This indicates that a saturation point exists, after which obtaining a marginal reduction of PFD_{avg} (i.e. improvement of safety integrity) involves a disproportionate cost. This situation could be only justified to meet a requirement to achieve some specific low PFD_{avg} value.

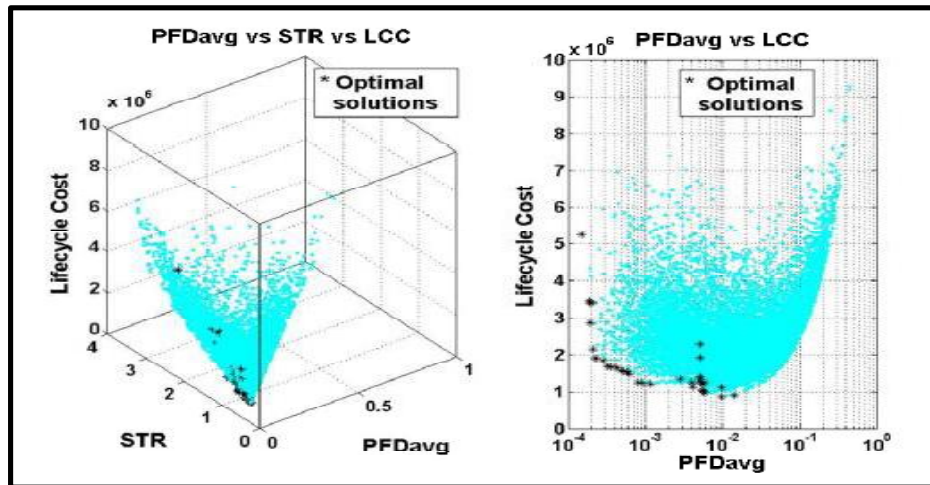


Figure 3.10. Result from one single run optimization, Modality 1

The plots in pairs of the four objectives (result form Modality 1) are shown in Figure 3.11. Some comments can be drawn from the pair wise relationships between objectives:

- PFD_{avg} and LCC are conflictive objectives in the Pareto-optimal front (see also Fig. 3.10). This is made possible by the fact that the contribution of the cost of hazards (i.e. catastrophic loss cost, C_{ACC} in Eq. (3.10), which is a fairly large figure) is not significant once the safety system is introduced. This is because when C_{ACC} is multiplied by the PFD_{avg}

(which is $\ll 1$) its influence is reduced drastically. Thus, this cost does not drive the optimization process any more. Therefore, it can be said that to meet one specific SIL requirement there are additional costs for the plant.

- PFD_{avg} and STR have points where their relationship is conflictive and points where this is neutral (no change in STR as response to a change in PFD_{avg}). This difference of behaviour can be explained in two parts: Firstly, the conflictive relationship is verified in the design changes. Whenever the optimizer changes any of the design variables (e.g. redundancy), a lower PFD_{avg} produces a higher STR, which is the normal relationship in parallel redundancy. The sections of the optimal front where reductions in PFD_{avg} do not cause increments in STR correspond to those where the optimizer does not change any design variable but rather reduces the Test Interval. This can be seen for example in Table 3.3 (Modality 3) and Figure 3.14 for solutions 4 to 7. In this group of solutions the system's design is fixed, and the only changing variable is the TI, affecting all variables excepting the STR. This suggests that the TI does not affect the STR, which needs further investigation.
- The trend of the STR vs. LCC relation suggests that they are generally not conflictive objectives: Increments in STR raises the LCC, suggesting that the cost of production losses by spurious trips is a significant burden. However, in a few cases increments of STR actually goes along with some cost reduction. This is possible when the optimizer changes several variables simultaneously at once, achieving a combination where the increment in STR (caused by higher redundancy) does not affect the LLC (for instance between solutions 10-11 and 15-16, see Figure 3.14) because the reduction in costs, other than loss of production, becomes dominant.
- PFD_{avg} and Unavailability are generally non-conflicting objectives up to a minimum point. However, in our zone of interest ($PFD_{avg} < 2 \times 10^{-4}$), they become conflicting objectives due to the growing unavailability by safe failure (see the STR in Table 3.3, Modality 3).
- The graph of Unavailability vs LCC shows the same conflictive behaviour as the plot PFD_{avg} vs LCC (since PFD_{avg} generally is the dominant factor of unavailability).
- A similar situation happens between the graphs of the couples Unavailability-STR and PFD_{avg} -STR: they show analogous relations.

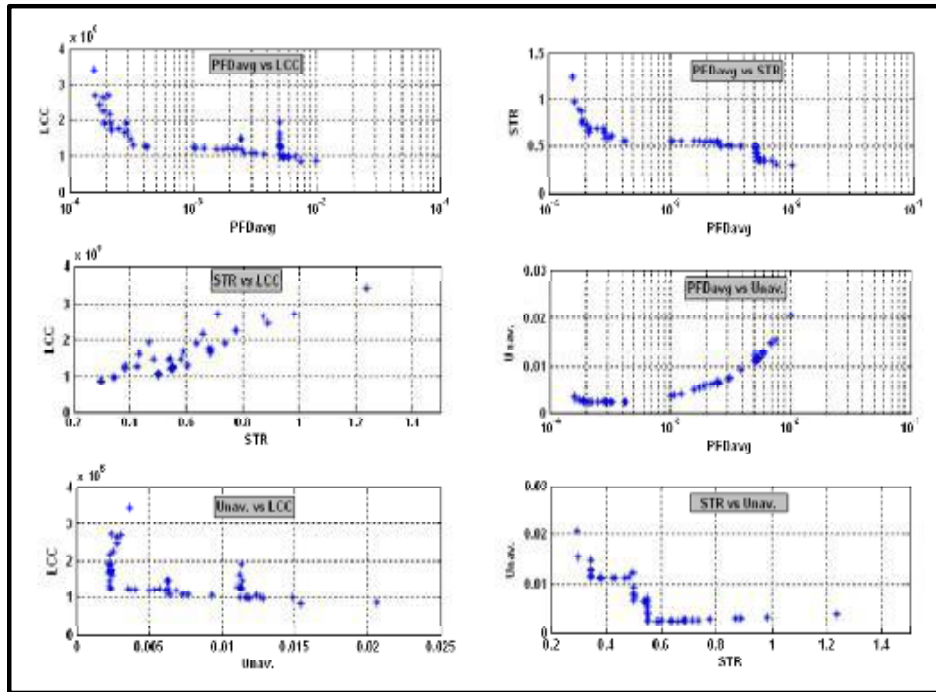


Figure 3.11. Pair wise plots of results from Modality 1

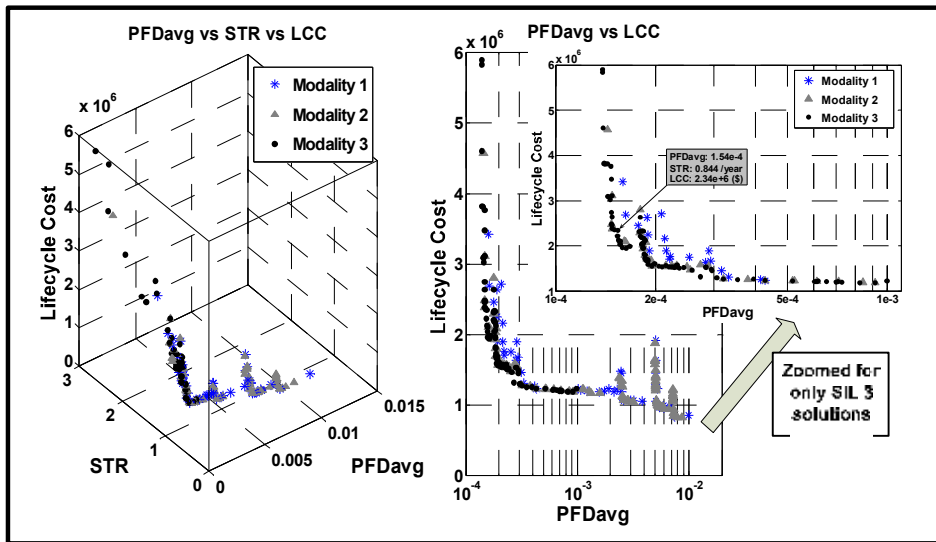


Figure 3.12. Comparison of results from all three modality approaches

Figure 3.12 shows the optimal solutions obtained from running the three different optimization program modalities described above. Notice that the solutions from Modality 1 are generally dominated by the solutions from the other two modalities. Modalities 2 and 3 seem to converge approximately in the same Pareto-optimal front. However, Modality 3 shows a slightly more

uniform distribution along the front in the SIL 3 region (i.e. $PFD_{avg} < 1 \times 10^{-3}$). Additionally, Modality 3 gave a set with many more SIL 3 solutions than the other two approaches (81 solutions, against 19 and 36 solutions from the other two). Therefore, it can be said that for seeking a specific SIL level Modality 3 is the best strategy.

Table 3.3 presents a selection of solutions from the three modalities that achieved SIL 3 by PFD_{avg} . Remember from Section 1.5 that to meet a specific SIL value requires both achieving a specific PFD_{avg} and meeting the necessary architectural constraints. Also notice that the row for the best solution from each modality is grey-highlighted in the table. From the optimal set given by Modality 1, solution number 1 was the only one that meets all the requirements (including architectural constraints), and from the set of Modality 2 solution 4 was the one that met the both the architectural constraints and PFD_{avg} requirements with the lowest LCC, STR and unavailability.

A set of 35 solutions from modality 3 are shown in Table 3.3 (out of a total of 81). It is evident that Modality 3 gave many more SIL 3 solutions in the optimal front than the other two modalities, which gives more options for decision making, especially for lower LCC and STR values. Figure 3.12 presents an inbox zooming up the SIL 3 region of the solution space. The situation in the Pareto-optimal front of solution 21 from Modality 3, the one chosen as the best one, is indicated in this figure. All in all, the best solution achieved by Modality 3 is better in all the objectives than the one given by the Modality 1 approach.

An example of the parallel coordinates graph shown in Figure 3.13 for the results of the Modality 3 approach. Observe the crossing lines between PFD_{avg} and STR, which manifest a conflictive relationship. Lines between STR and LCC are mostly not criss-crossing, confirming that the prevalent relation between them is not conflictive.

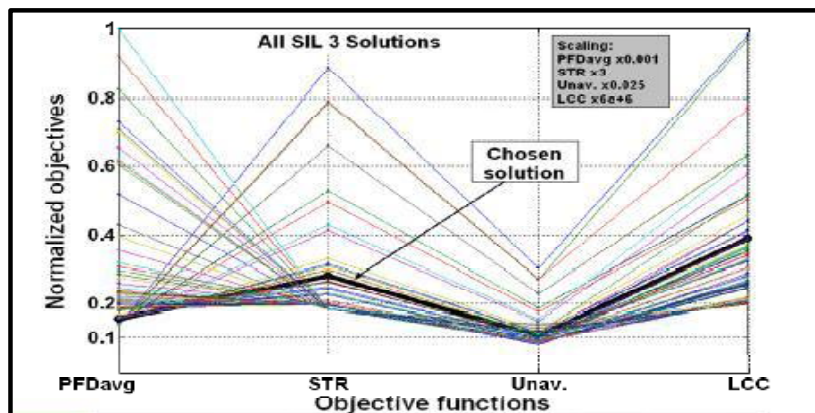


Figure 3.13. Parallel coordinates graph Modality 3 approach with final chosen solution

Table 3.3. Some optimization results that achieved SIL 3 by PFD_{avg}

S	SUBSYSTEMS CONFIGURATION											RESULTS					
	PT				LS				FC			PFD_{avg} ($\times 10^{-4}$)	STR (/year)	Unav	SIL Arch	LCC ($\times 10^6$)	
	T	R	β	TI	T	R	β	TI	T	R	β						TI
Modality 1																	
1	2	3	0.02	1	2	3	0.02	16	2	4	0.02	1	1.5946	1.23790	0.0037028	3	3.4212
2	2	2	0.02	1	2	3	0.02	12	2	3	0.02	1	1.6270	0.98291	0.0030166	2	2.6864
3	2	4	0.02	1	3	2	0.02	2	2	3	0.02	1	1.7711	0.88738	0.0027809	2	2.4550
Modality 2																	
1	4	3	0.02	1	2	3	0.05	2	2	4	0.02	1	1.4359	2.26720	0.0064895	4	4.5673
2	2	3	0.02	1	2	2	0.02	1	2	3	0.02	1	1.4743	0.94519	0.0028772	3	2.4788
3	2	4	0.02	1	4	3	0.02	1	2	3	0.02	1	1.4836	0.93002	0.0028376	3	3.1105
4	2	3	0.02	1	4	2	0.02	1	2	3	0.02	1	1.4840	0.84434	0.0026030	3	2.3817
5	2	2	0.02	1	2	2	0.035	5	2	3	0.02	1	1.6085	0.89493	0.0027712	2	2.1024
6	2	2	0.02	1	2	2	0.035	6	2	3	0.02	1	1.6230	0.89493	0.0027741	2	2.1012
Modality 3																	
1	4	4	0.02	1	2	3	0.02	1	2	4	0.02	1	1.3886	2.66390	0.0075685	4	5.8950
2	4	3	0.02	1	2	4	0.02	1	2	4	0.02	1	1.3887	2.35910	0.0067335	4	5.8323
3	4	4	0.02	1	2	2	0.02	1	2	3	0.02	1	1.3909	2.37120	0.0067675	3	4.6023
4	4	3	0.02	1	2	2	0.02	2	2	3	0.02	1	1.3993	1.97980	0.0056967	3	3.8216
5	4	3	0.02	1	2	2	0.02	3	2	3	0.02	1	1.4076	1.97980	0.0056984	3	3.8154
6	4	3	0.02	1	2	2	0.02	4	2	3	0.02	1	1.4160	1.97980	0.0057001	3	3.8123
7	4	3	0.02	1	2	2	0.02	6	2	3	0.02	1	1.4328	1.97980	0.0057035	3	3.8092
8	4	2	0.02	1	2	2	0.02	2	2	3	0.02	1	1.4408	1.58830	0.0046360	3	3.0895
9	4	2	0.02	1	2	2	0.02	3	2	3	0.02	1	1.4491	1.58830	0.0046377	3	3.0833
10	4	2	0.02	1	4	2	0.02	2	2	3	0.02	1	1.4617	1.48750	0.0043641	3	3.0221
11	2	4	0.02	1	2	3	0.02	1	2	4	0.02	1	1.4718	1.28680	0.0038122	4	3.7646
12	2	3	0.02	1	2	3	0.02	1	2	4	0.02	1	1.4720	1.23790	0.0036782	3	3.4736
13	2	4	0.02	1	2	2	0.02	1	2	3	0.02	1	1.4741	0.99412	0.0030112	3	2.7418
14	2	3	0.02	1	2	2	0.02	1	2	3	0.02	1	1.4743	0.94519	0.0028772	3	2.4788
15	2	3	0.02	1	2	2	0.02	2	2	3	0.02	1	1.4826	0.94519	0.0028789	3	2.4601
16	2	4	0.02	1	4	3	0.02	1	2	3	0.02	1	1.4836	0.93002	0.0028376	3	3.1105
17	2	4	0.02	1	4	2	0.02	1	2	3	0.02	1	1.4838	0.89328	0.0027370	3	2.6406
18	2	3	0.02	1	4	2	0.02	1	2	3	0.02	1	1.4840	0.84434	0.0026030	3	2.3817
19	2	3	0.02	1	4	2	0.02	2	2	3	0.02	1	1.5035	0.84434	0.0026070	3	2.3568
20	2	2	0.02	1	2	2	0.02	1	2	3	0.02	1	1.5372	0.89625	0.0027612	2	2.2217
21	2	3	0.02	1	4	2	0.02	4	2	3	0.02	1	1.5431	0.84434	0.0026150	3	2.3444
22	2	2	0.02	1	4	2	0.02	1	2	3	0.02	1	1.5468	0.79541	0.0024870	2	2.1287
23	2	2	0.02	1	4	2	0.02	2	2	3	0.02	1	1.5663	0.79541	0.0024909	2	2.1038
24	2	2	0.02	1	4	2	0.035	1	2	3	0.02	1	1.5674	0.79485	0.0024890	2	2.0370
25	2	2	0.02	1	4	2	0.05	1	2	3	0.02	1	1.5879	0.79429	0.0024909	2	1.9668
26	2	2	0.02	1	4	2	0.05	2	2	3	0.02	1	1.6359	0.79429	0.0025005	2	1.9420
27	2	2	0.02	1	3	2	0.02	1	2	3	0.02	1	1.6742	0.78951	0.0024970	2	1.9873
28	2	4	0.02	1	2	2	0.02	1	2	2	0.02	1	1.7880	0.78809	0.0025392	2	2.3159
29	2	4	0.02	1	4	3	0.02	1	2	2	0.02	1	1.7975	0.72399	0.0023656	2	2.6347
30	2	4	0.02	1	4	2	0.02	1	2	2	0.02	1	1.7977	0.68724	0.0022650	2	2.2091
31	2	3	0.02	1	4	3	0.02	2	2	2	0.02	1	1.8167	0.67505	0.0022355	2	2.3395
32	2	4	0.02	1	4	2	0.02	2	2	2	0.02	1	1.8171	0.68724	0.0022690	2	2.1843
33	2	3	0.02	1	2	2	0.02	5	2	2	0.02	1	1.8216	0.73915	0.0024120	2	2.0365
34	2	3	0.02	1	2	2	0.035	3	2	2	0.02	1	1.8305	0.73783	0.0024094	2	1.9608
35	2	2	0.02	1	3	2	0.02	2	2	3	0.02	1	1.8341	0.78951	0.0025309	2	1.9540

S=Solution, T=Type, R=Redundancy, β =Beta factor, TI=Test interval
 SIL arch = SIL achieved according with the system's architectural constraints

It can be said that for a general approach, Modality 2 gives better results since it comprises a larger range of solutions (i.e. more diversity) with a uniform distribution along the Pareto-optimal front. The Modality 3 approach is more suitable if some specific goal is being sought (for example SIL 3 compliance in this case).

Figure 3.14 shows the plot of the entire Pareto-optimal set given by Modality 3. The upper plot presents only the objective function values, the rest shows a single plot for the decision variables of each subsystem. Notice that the graphs are arranged according to the decreasing values of the $PF_{D_{avg}}$. Thus, the numbering of solutions runs in inverse order. Also note that from this complete set, only solutions 1 to 35 are detailed in Table 3.3.

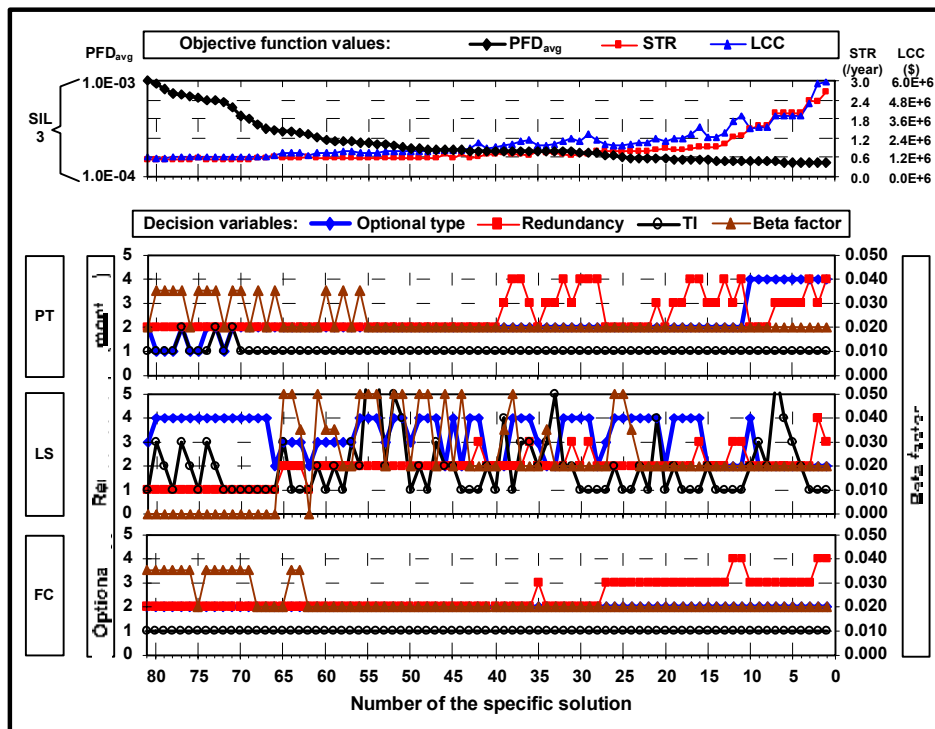


Figure 3.14. Complete Pareto-optimal set given by Modality 3

The following remarks can be made based on Table 3.3 and Figures 3.11-3.14:

- Overall, Modality 3 gave better results than Modality 1. However, while (in Modality 3) lower values have been achieved for $PF_{D_{avg}}$ (than in Modality 1), higher values have been obtained for the STR (see Table 3.3). This is a situation that the optimizer cannot avoid, since it is a condition inherent to the problem and the equipment provided.

- Also remember that the goal was to achieve a maximum PFD_{avg} of 1.7×10^{-4} . Table 3.3 shows that only two and six solutions from Modalities 1 and 2 respectively met this requirement. Notice that amongst those eight solutions, some achieve only SIL 2 by architectural constraints (which column's respective sections are highlighted red in Table 3.3). Therefore, only five solutions from these sets can be chosen as valid. If we had to choose one best solution from each set, they would be solutions 1 and 4 respectively.
- As with Modalities 1 and 2, only a fraction of the set of solutions given by Modality 3 achieving SIL 3 by PFD_{avg} can be actually used as a solution to the problem, since the architectural constraints do not permit claiming a SIL higher than 2 for most of them (e.g. see red-highlighted column's section in Table 3.3). It is worth mentioning that if type A subsystem components were used instead of type B (with the same reliability specifications) for the design of the safety system, the proportion of solutions that could claim a SIL level higher than 2 by architectural constraints would be larger, since IEC 61508 poses less strict constraints for type A components. In consequence there would be more solutions in the Pareto-optimal front to choose from, which could give better results in terms of the four optimized objectives, including cost. For instance, if we had the same components considered in this problem but they were type A, some better solutions could be picked from Table 3.3 (see solutions 22 to 27 of Modality 3, which meet the maximum PFD_{avg} requirement).
- Solution 21 from Modality 3 has been selected as the best one (highlighted in grey in Table 3.3). This is the one that meets all the requirements and achieves SIL 3 by architectural constraints. Note that this solution gives better overall results in all the four objectives than solution 1 of Modality 1, and lower LCC with the same STR than solution 4 of Modality 2.
- Looking at Figure 3.14, it can be confirmed that PFD_{avg} and STR show mostly a conflicting relation, although in some parts decrements in PFD_{avg} actually do not increase the STR, or require very short increments. Solutions 1-35 however, do evidently cause increments in STR when reducing PFD_{avg} (what can be confirmed analyzing the numbers in Table 3.3). Therefore, considering this and the comments previously drawn, it can be concluded, that PFD_{avg} and STR have a relationship that is between neutral (i.e. the change of one do not cause a change on the other, which happens when only TI is changed) and conflictive (changes in design variables). The conflictive relation is dominant for lower levels of PFD_{avg} .

- Notice from Table 3.3, that amongst the solutions achieving a SIL 3, the more convenient optimal solutions use the optional-type 2 for transmitters and valves (which according to Table 3.1 have low failure rate, and high diagnostic coverage and cost). Observe in Table 3.3 that transmitters optional-type 4 are used for solutions with a PFD_{avg} lower than 1.47×10^{-4} , which nevertheless increments the STR and LCC even further (which can be easily seen in Fig. 3.14).
- The β factor values chosen in solutions 1-35 are the lowest possible for redundant systems ($\beta=0.20$) in all the subsystems. This indicates that electrical separation for field instruments (PT and FC) and electrical separation plus physical segregation for LS, give the best results. In solutions 35-81 the β values are not constantly the lowest possible ones, but those solutions do not achieve a PFD_{avg} as low as the established goal.
- In addition, testing the equipment as frequently as permitted by the problem (i.e. every month, see Fig. 3.14) is the more selected option. The fact that the chosen field instruments are the second most expensive option available and that they must be tested very frequently indicates that perhaps a equipment with better specifications should be sought to permit a lower Test Interval. This chapter assumes perfect testing. However, remember that imperfect testing can overlook potential problems, and even induce them in extreme cases.
- Observe in the set of solutions of Modality 3 that meet the PFD_{avg} goal (solutions 1-35) that the main trade-offs are in the redundancy variable. In general, it can be noticed that changes in redundancy are quite influential (observe that when passing from redundancy=1 to >1 the adverse effects of CCF are also introduced).
- In addition, notice that for the complete set of solutions, the main trade-offs are for the Logic Solver subsystem's design rather than the field instruments. Observe that the graphs of the PT and FC subsystems are relatively stable compared with the LS graph. Comparing to the field instruments (PT and FC), the components available for the FS subsystems have lower total failure rates, much higher diagnostic coverage and a very high cost. They also have more options for CCF reduction. This difference seems to provide the optimizer with more room to manoeuvre. Observe that the PT and FC subsystems are practically stuck in $TI=1$ month and $\beta=0.2$. This could indicate that the CCF rates are quite high on those subsystems, and the optimizer has exhausted the resources for β reduction and dependability improvement by test. Therefore, it resorts to trading with the LS variables. This hypothesis is feasible if we compare PT against FC performance. Again, the PT components have

better reliability specifications, so the optimizer trades more with the PT variables than with the FC variables, which seem to be rather static. Also note that the majority of solutions implement the two best-diagnosed and most expensive LS optional-types (2 and 4), similar to the situation for the PT. It is notable that equipment with good diagnostic coverage is generally chosen. Again, it can be said that equipment choices available may be not appropriate for the selected application, since a high Test Interval is required (once per month). In addition, the STR seems to be quite high for a safety system, which reinforces this hypothesis.

3.9. CONCLUDING REMARKS

This chapter introduced the initial case of multi-objective optimization of SIS based on IEC 61508 requirements and using GAs as optimization method. It demonstrates the fundamental applications of concepts detailed in previous chapters, integrating it for achieving practical results.

First of all, as demonstrated by the Modality 2 approach, it was seen that feeding back optimal solutions for previous runs of the GA as initial population of the next run can improve the Pareto-optimal set obtained (Modality 2 gave overall better results than Modality 1). What is more, feeding back such a population in a portion which objective values are closer to a specific goal (like the specific SIL required in the presented problem) can actually guide the search towards that goal (Modality 3 gave a much larger optimal set around that goal).

It was demonstrated that introducing the safety system and making an initial improvement of its safety integrity is economically convenient, since it actually reduces the initial LCC. Besides, any solution included in the Pareto-optimal set has far better trade-offs (e.g. must lower costs) than the initial solutions.

The objectives to optimize included PFD_{avg} , STR, Unavailability and LCC. It has been seen that introducing Unavailability as optimization objective does not provide any significant improvement to the method since this measure is implicitly contained in the PFD_{avg} and STR. It can be, therefore, omitted in future optimizations.

PFD_{avg} is consistently in conflict with the system LCC in the Pareto-optimal front. This means that achieving any improvement of safety integrity will have an additional cost. In addition, there is a saturation point where any marginal improvement has a significantly large cost increment. Regarding its relationship to STR, PFD_{avg} is sometimes in neutral relation and

sometimes in conflict. Nevertheless, they are not in any case harmonious objectives. It is matter of further investigation to determine how this relationship behaves. In contrast, STR and LCC do not show a consistent conflictive relationship.

The architectural constraints imposed by IEC 61508 (see Section 1.5.) can actually decrease the SIL that a certain design can achieve. All solutions presented in Table 3.3 did achieve SIL 3 by their PFD_{avg} value. However, many of them (highlighted in red) are limited to claiming a lower SIL by the architectural constraints.

It was seen that the optimizer gives preference to components with better reliability specifications (like lower failure rates and higher diagnostic coverage) in spite of their higher acquisition cost. Reduction of CCF is also a highly favoured factor. It has been observed that the optimizer consistently chooses the lowest β factor values possible. CCF is a very significant negative influence on the system dependability values. An optimization method addressing this problem is presented in the next chapter.

The optimizer also selected the highest test frequency available as TI option. It has been seen that increasing testing frequency seems not to have influence over STR, and only reduces PFD_{avg} . It is, however, a matter of further research to investigate whether this frequent testing can have some negative effects not being considered here. This problem is approached in Chapter 5.

The parallel coordinates graphs suggested by Fonseca & Fleming (1993) is actually a useful tool that can assist the decision making problem. However, for a more generic analysis other kind of graphs (e.g. Figs. 3.12 and 3.14) were more useful.

The case study presented in this chapter has permitted verification of the relationship of PFD_{avg} versus STR in parallel redundancies. Incrementing parallel redundancy with the aim of reducing the PFD_{avg} increases the STR because every new component added is a new chance of safe failure. In optimization of design this can be mitigated by other alternatives, such as choice of different components (with lower safe failure rates) or reduction of CCF. Results in Figure 3.14 showed, especially at the right end of the graph, that reducing the PFD_{avg} by solely increasing redundancy necessarily raises the STR. This effect can be counteracted by using Moon voting redundancy rather than simple parallel, in the search of achieving a better balance PFD_{avg} -STR. This is analyzed in detail in Chapter 6.

It can be said that the optimization method has been successfully implemented in its initial version. The results obtained by the optimizer also allowed an analysis of system design in terms of dependability and cost to be undertaken.

In this chapter a specific goal of PFD_{avg} has been set for the optimization in order to explore how the search can be guided by the formation of the initial population. In practise, the acceptable limits of PFD_{avg} can be set between determined bands (i.e. compliance with a determined SIL level) or it can be a hard constraint. The latter is the case when a specific risk reduction value has to be achieved, which is a frequent requirement. In this thesis, which intends to implement a SIS optimization methodology, the search is not constrained with the aim of developing the ability of the optimizer to explore the whole search space and the relationship among the different objectives along the entire Pareto-optimal front. Therefore, the case studies in subsequent chapters do not present the desired level of PFD_{avg} as a hard constraint.

CHAPTER 4

Optimization of Design with Diverse Redundancy

This chapter analyzes the optimization of Safety Instrumented System design introducing diverse redundancy in the subsystems as a defence against Common Cause Failure (CCF). CCF can have a significant influence on a system's dependability, and it can even be the dominant factor affecting it. Among several design measures, the use of diverse redundancy is an effective defence against CCF. However, it is a topic barely researched in safety system optimization. This chapter primarily presents an introduction to the CCF phenomenon, the quantification of the CCF share over the total failure rate of a component and its modelling at system level. A novel index for quantification of the level of diversity in a system is introduced. This permits comparison of the diversity index of a particular system design against its performance, and thus to assess its benefits in relation to rival designs. Following a similar structure to the previous chapter, a methodology for modelling dependability (adaptable to the changing demands of the optimization activity) and Lifecycle Cost is presented. Then a case study exemplifies the implementation of the multi-objective optimization program. The present chapter is aimed at demonstrating how the consideration of diverse redundancy can provide even more flexibility for RAMS+C based decision-making as a consequence of its positive impact, not only on system safety integrity, but also on Lifecycle Cost.

4.1. COMMON CAUSE FAILURE

4.1.1. The phenomenon of CCF

Common Cause Failure is the term used to define the event of the simultaneous failure of several components due to the same cause. According to CCPS (2000), there are three conditions for an event to be considered a CCF:

1. Multiple components are failed.
2. The failures must be simultaneous. This means *“sufficiently close in time to result in failure to perform the safety function required of the multiple [components]”*.
3. The cause of the failure must be the same for all the failed components.

The phenomenon of CCF deserves an important consideration because it can affect considerably the dependability of the system. Given that safety systems consistently employ redundancy in their architectures to implement the safety function with the required safety integrity, CCF

becomes a fundamental issue since it can negate the entire benefits of redundancy when failing all the redundant components simultaneously. It therefore can increase considerably the PFD of a system. For high-integrity systems (which are normally implemented with components with very low failure rates) CCF can particularly become the dominant factor in the system unavailability. Since CCF affects the failure rates of the components, it also influences considerably the system's STR.

4.1.2. CCF modelling

Several different models have been proposed for modelling CCF. Mosleh et al. (1988) presented an account of the main parametric methods used for quantification in safety and reliability studies. He categorized the parametric models as shock or non-shock models. Non-shock models include the best known models for safety and reliability studies: β Factor, Multiple Greek Letter and the Alpha Factor methods. For shock models, the binomial failure rate model is mentioned. A detailed account of those models is presented in Appendix A.

The *β factor model* (Mosleh et al., 1998) is a single-parameter model that assumes that the total failure rate of a component is the sum of independent and common cause failures (Eq. (4.1)), and that the fraction of the common cause failure rate over the total is the β factor (Eq. (4.2)). This can be interpreted as a fixed fraction β of the total failure rate of a component (belonging to a group of components) that can be attributed to a common cause event. It also assumes that the common cause event will fail all the components of that group.

$$\lambda^T = \lambda^N + \lambda^C \quad (4.1)$$

$$\beta = \frac{\lambda^C}{\lambda^N + \lambda^C} \quad (4.2)$$

Given that the total failure rate of a component is the sum of the independent and the CCF (dependent) failure rates (Eq. (4.2)), it is possible to determine the independent failure fraction (also called normal failure, hence the N superscript) and the common cause (dependent) failure fraction of the component based on the total failure rate and the estimated β factor (Eqs. (4.3) and (4.4) respectively).

$$\lambda^N = (1 - \beta) \cdot \lambda^T \quad (4.3)$$

$$\lambda^C = \beta \cdot \lambda^T \quad (4.4)$$

Between all models for CCF, the β -factor model remains as the most widely applied due to its simplicity and traceability to real design and operating conditions. Mosleh et al. (1988) expressed that considering the state of data involving large uncertainties, the impact of choosing

a particular model over another is not highly relevant (unless sound statistically significant data were available). They added that the β factor model provides results with reasonable accuracy for redundancy up to four (albeit slightly conservative), although for higher levels of redundancy the method is still applicable, but more conservative, and more sophisticated models would be advised. Smith (2005) points out, in addition, that the Multiple Greek Letter (MGL) model is yet too sophisticated to be supported by field data until more detailed information is available. The author of this thesis considers that the same consideration is applicable to the other multi-parametric methods. Although some guidance to estimate the parameters of multi-parametric models, it is still difficult to relate this estimation to the perceived causes of CCF and specific plant design and operation conditions.

It is a fact that the original β factor model (Fleming, 1975; as cited in Mosleh et al., 1988) is a rather basic estimation of CCF. However, due to its simplicity and practicality it has survived as the most used model for real life applications. This has been made possible, however, by further refinements practised to the original model. Some of them are presented as new models, but the author considered they are extensions of the β factor model. The partial β factor model (as described by Smith, 2005) splits the total β factor into several contributions that influence the CCF (i.e. similarity, separation, complexity, analysis, procedures, training, control and test). This is equivalent to having one “partial β ” for each contribution, which are summed up to determine the final β factor. This allows estimation of the β factor using engineering judgment based on the actual design and operational conditions. Smith (2005) developed a new extension in the Betaplus model, with the aim to make the model more objective and maximize the traceability in the estimation of β . It takes into account important features relevant to SIS, such as equipment type and diagnostic coverage. The method for quantification of CCF suggested by IEC 61508 is based on these models, as will be detailed in the next section.

The β factor model does not account for differences in the number of redundant components and the MooN voting architecture (where $M < N$). The β factor is the same regardless of these features. The PDS Method (Hauge et al., 2006a) introduced a new development to modify the basic β factor (relevant to two components in parallel) by an additional factor C_{MooN} dependent on the voting configuration (Eq. (4.5)). The proposal has been developed from Hokstad & Corneliussen (2004), Hokstad (2004), and Hokstad et al. (2006), where it is called the *Multiple β factor model*. A salient feature of the proposition made in the PDS method is that suggested values of the C_{MooN} factors are given for quantification of $PF_{D_{avg}}$ and STR.

$$\beta' = \beta_{MooN} = \beta \cdot C_{MooN} \quad (4.5)$$

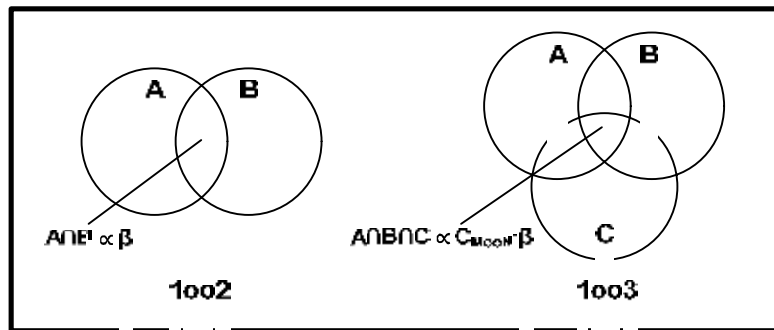


Figure 4.1. Modification of the β factor for a 1oo3 system

The intersection of events shown in Figure 4.1 represents the fact that the occurrence of failure of multiple components by a common cause is not probabilistically independent. As can be seen, the intersection is smaller for three (or bigger) number failed components. The suggested value for a 1oo3 voting, for example, is a third of this intersection for only two components ($C_{MooN}=0.3$). This is the logic behind the SINTEF's C_{MooN} modification factor.

4.1.3. Quantification of CCF at system level

A seminal work presenting a complete methodology for treating CCF in safety and reliability studies is Mosleh et al. (1988). A summary was presented in Mosleh (1991). Mosleh et al. (1988) identified the three essential factors to consider when analysing CCF:

- The root cause of the failure, which is the event or factor that leads to CCF
- The coupling mechanism, which permits that one single root cause affects multiple components. Coupling is what makes a real difference between independent and common cause failures (CCPS, 2000). This reference details six different types of coupling: *common support system*, *common hardware*, *equipment similarity*, *common location*, *common internal environment* and *common operating or maintenance staff and procedure*.
- The lack of defences against CCF mechanisms. Defences can approach the elimination of root causes or coupling mechanisms. These are of engineering or operational nature. Examples of defences are separation and segregation of equipment, design review, test and maintenance procedures, training of operation and maintenance staff, etc.

Mosleh et al. (1988) established that the couplings that may lead to CCF must be identified when developing safety and reliability analysis, and that the probability of these events must be quantified in order to have a realistic assessment. CCF events can have important contributions to the overall unreliability of a system, since they become basic failure events, and can even become a dominant factor of the overall failure probability of a redundant system. Section A.2 lists the steps of the procedural framework for CCF analysis defined in Mosleh et al. (1988).

IEC 61508 Part 6 provides a methodology for estimating the β factor for the specific application being analysed. Although the standard does not mention any direct reference, it seems to be based on the Partial Beta model and resembles very closely the Betaplus model mentioned above (both explained in Smith, 2005).

The methodology suggested by IEC 61508 bases the estimation of the β factor on determining which defence measures against CCF have been put in place, which fall into one of these three categories: reduction of overall failure rates, maximization of independence of channels, and revelation of non-simultaneous CCF. The measures are thus enlisted by the type of defence measures: 1) separation/segregation; 2) diversity/redundancy; 3) complexity/design/application/maturity/experience; 4) assessment/analysis and feedback of data; 5) procedures/human interface; 6) competence/training/safety culture; 8) environmental control; 9) environmental testing.

This methodology conveniently extends the basic model so that it can be determined on the basis of the specific system design and plant conditions. It considers the technological differences between logic subsystems and field instrumentation (sensors and final elements) and their dissimilar working conditions. It also takes into account the difference between factors that are enhanced by the diagnostic coverage and those that are not.

The standard provides tables that allow determination of a score for each defence measure of the mentioned categories put in place (based on engineering judgement). Tables provide the final β^U factor that corresponds to undetected failures, and β^D for detected failures based on those scores. The tables provide separate scoring values for logic solver and for field instruments. The reader is referred to IEC 61508 part 6, Annex D to see the mentioned tables.

For the sake of simplicity, IEC 61508 makes the same assumption of the basic β factor model that once a CCF event takes place it affects all the channels in the redundant subsystem regardless of the level of redundancy. This approach does not distinguish between different schemes of redundancy or voting logic. However, the approach can be complemented with the formulation made by Hauge et al. (2006a), shown in Eq. (4.5), to include the difference in CCF for different Moon architectures.

There are two different approaches for incorporating CCF into Fault Tree Analysis and other combinatorial methods: implicit and explicit methods. The description given by Vaurio (1998) is that in the explicit method the CCF is modelled as basic events in the fault tree. These events are repeated as input to all components they affect. These events are preserved by Boolean

reduction. On the contrary, in the implicit method only independent basic events are modelled in the system logic initially. The terms of the probability expression obtained after Boolean reduction are evaluation in such a way that the contribution of CCF is included. Vaurio (2002) recognized that the implicit method may require a large number of terms to be quantified manually, and that truncation of the system equation in large systems can carry the possibility of losing important terms. He proposed a methodology for transformation of implicit into explicit models, given the reasons previously mentioned and that the former can handle automatically independent events. It is therefore reasonable to say that explicit methods are more flexible and perhaps more powerful.

The CCF basic events generated in explicit methods can be subject to Boolean reduction to represent only the Minimal Cut Sets (MCS). The simplest example is the CCF of a double parallel redundant system shown in Figure 4.2. All the components that would be failed by a common cause are included in one single group. These are usually the identical components in a redundant arrangement. This group is then represented by a fault tree, the top event of which is originated by an OR gate whose inputs are the simultaneous independent failure events of the components plus the event that fails all components by common cause.

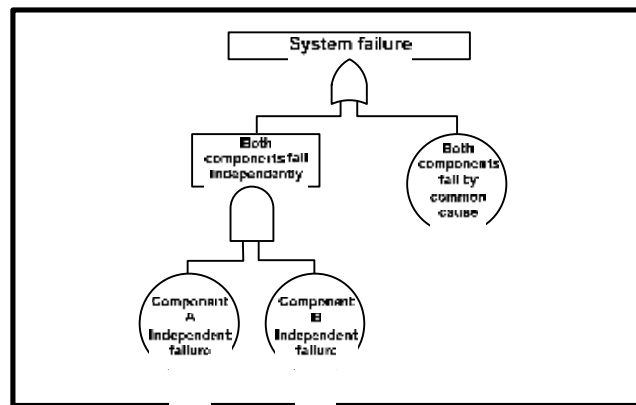


Figure 4.2. Modelling of CCF for a two-component system

Vaurio (2003) has developed analytical equations that include CCF probabilities for quantification of the average unavailability of redundant standby safety systems for up to four components. It includes simultaneous, sequential and uniform staggered testing. Quantification of CCF is based on the General Multiple Failure Rate model (a general formulation of the Alpha factors and the MGL models (Vaurio, 1994)). This work in Vaurio (2003) is a continuation of previous work by the same author (Vaurio 1994, 1995b, 1998, 2002). The resulting equations however do not take into account separation of detected and undetected failure modes, and

therefore it does not consider the effects of diagnostic coverage. They are also tied to a uniform distribution of the test events in case of staggering testing.

In this work the CCF is quantified using the β factor model. As can be concluded from the previous discussion, the main advantages are its simplicity and traceability to actual system design and operation conditions. Different from multi-parameter models, it only requires estimation of the β factor and to have the total failure rate of the component. Multi-parameter models are more sophisticated, but without a practical method for estimation of the various parameters its applicability is not feasible. The β factor model also presents a similar advantage over shock models, where the various shock events and the conditional probability of CCF subject to all these shocks would have to be estimated. In addition, if the method given by IEC 61508 is complemented with the framework given by Mosleh et al. (1988) and the modification formulated by Hauge et al. (2006a), the model can be considered well developed for incorporation into the modelling and optimization studies in this work. This model is, to the best of the author of this thesis' knowledge, the only one currently used for the assessment of SIS based on operational data so far.

4.1.4. Optimization considering CCF

Even though CCF is a very influential factor in system unavailability, very few optimization works consider it. Total independence is, in contrast, usually assumed between failure events of redundant components. Bai et al. (1991) presented an early optimization study of the number of redundant components n in k -out-of- n systems based on reliability at the time of minimizing the mean cost rate, using predetermined CCF rates. Pham (1993) determined system reliability against number of spares on the search for their optimal value in a TMR-plus-spares system, considering predetermined values of CCF and diagnostic coverage. Coit & Smith (1996a) addressed the optimization of redundancy allocation with component selection for a series-parallel system (including k -out-of- n :G subsystem redundancy), where components of different types could be mixed, using a GA. They solved reliability maximization given cost and weight constraints, and cost minimization given reliability and weight constraints. In a subsequent work, Ramirez-Marquez & Coit (2007) addressed the problem by non-linear programming permitting mixing diverse components in series-parallel systems. They estimate CCF using the discrete shock model presented in Kvam & Martz (1995). Different cases with various CCF values are analyzed for a series system with five parallel-redundant subsystems. Yu et al. (2007) also addressed reliability optimization for a redundant system with failure dependencies, modelling different redundant dependencies (independence, weak, linear and strong).

4.2. DIVERSITY

4.2.1. The role of diversity against CCF

It has been mentioned in Section 1.1.3 that coupling mechanisms have a large influence in the difference between independent and common cause failures. One of those mechanisms has been defined as “equipment similarity”. Equipment that is similar or, even more, identical can be expected to be affected by the same causes. Goble (1998) establishes three basic design defence rules against CCF: physical separation, diversity and making the design more robust for higher strength (“*ruggedization*”). Diversity means that units that have different design or construction are put together in a redundant arrangement. This tends to eliminate the susceptibility of the units to the same common cause of failure. Notice that one of the defence categories that IEC 61508 uses for quantification of the β factor is diversity/redundancy.

It was seen in Chapter 1 that fault tolerance, the capacity of a system to prevent single failures escalating into system failures, is usually achieved by some form of redundancy. This involves the use of extra equipment or parts that the system would not normally need to perform its function, but are used to tolerate potential faults. Plain identical hardware redundancy may, however, introduce susceptibility to CCF, which in turn reduces drastically the benefits of redundancy. In consequence, CCF can be counteracted by implementing redundancy with components technologically diverse; i.e. diversity in redundancy. Observe that whilst identical redundancy aims to address the general problem of random failures, the technique of diversity is used for additionally tackling the particular problem of CCF. When implementing diversity, a function is implemented in different ways in the hope that the same fault is not present in two diverse technologies or techniques.

Although the concept of diversity as defence against dependent failures in redundant systems is well known, investigation of its benefits and strategies is a topic barely studied. For example, Littlewood & Miller (1989) approached the estimation of probability of failure of multi-version software system with forced diversity of design. Littlewood (1996) subsequently derived an analogous model for hardware forced diversity, in which it is expected that differently designed hardware items responds differently (i.e. probability of failure is different) to a particular operational environment (i.e. stresses). He concludes that “*the best design is the one that uses all the available types of components, but uses each as little as possible*”. Fischer & Piel (1999) made an analysis on the achievement of independence between redundant trains of computer-based safety instrumentation and control systems in the nuclear industry, focussing on the logic solver (i.e. the computer). They classified diversity as physical (different design) and functional (different operation principle). They analyze in which case the implementation of diversity is

convenient and under which modality. Interestingly, they remark that a reasonable way to implement physical diversity is the use of non-safety equipment to implement redundancy of safety functions. This is justifiable if the final safety (i.e. availability) performance meets the relevant targets. The case of implementing diversity with smart sensors (i.e. sensors with enhanced capabilities by a microprocessor) was approached by Meulen (2003). He recognizes the advantages of using diversity to counteract CCF among redundant systems using smart sensors, and analyzes in which ways it can be implemented. Nevertheless, he also recognizes that diversity can have some disadvantages, mainly of an operational nature (e.g. different types of equipment have to be maintained, which could potentially lead to maintenance errors).

This chapter presents an optimization of system design using diversity as a countermeasure against CCF. Different from the few approaches that have previously contemplated CCF in optimization (e.g. especially diagnostic coverage is omitted), this work considers all modelling detail relevant to safety systems. This work implements a practical approach that uses the β factor model for estimation of CCF (together with some enhancements mentioned above), that allows incorporation of engineering judgement and traceability to the real design and plant operating conditions. Notice that the simplicity of the β Factor model reduces the problem of getting CCF data for industrial equipment to the availability of suitable historical statistically significant failure rate data, in addition to the system's design and operation data.

IEC 61508-4 defines diversity as “*different means of performing a required function*”. It adds a note complementing that “*diversity may be achieved by different physical methods or different design approaches*”. In this thesis diversity is implemented permitting the mixture of technologically diverse equipment in redundant subsystems.

4.2.2. Diversity quantification

In order to analyze more effectively the effect of diversity on system dependability and cost a Diversity Index (DI) was formulated. The objective of this index is to reflect the relative level of both redundancy (*Redundancy/Components*) and diversity (*Technologies/Components*) to the total number of components. The first formulation made was the following:

$$DI_1 = \frac{1}{N_s} \sum_i^{N_s} \frac{Technologies_i}{Components_i} \cdot \frac{Redundancy_i}{Components_i} \quad (4.6)$$

Where N_s is the number of subsystems. Notice that the term *Redundancy* refers to the number of extra components not necessary for the execution of the safety function but added for the sake of subsystem dependability improvement (i.e. *Components-1*). The term *Technologies* simply refers to the number of different technologies used per subsystem.

The index given by Eq. (4.6) may be somewhat insensitive to changes in redundancy when this is changed simultaneously with diversity. For this reason the index was slightly modified into the index given by Eq. (4.7). The two indexes are numbered with a subscript just for the purpose of facilitating the analysis, made at the end of the case study.

$$DI_2 = \frac{1}{Ns} \sum_i^{Ns} \frac{Technologies_i \cdot Redundancy_i}{Components_i} \quad (4.7)$$

4.3. DESCRIPTION OF THE APPLICATION PROBLEM

The application problem is about redundancy allocation with component selection (discrete reliability allocation) with the option of diverse redundancy for a series-parallel system. Different from the study case of the previous chapter, this case is only about design optimization. In Chapter 3, where optimization of design was mixed with optimization of test frequency, it was observed that the optimizer consistently selected the highest possible test frequency. It is a matter of further research to verify that such a frequent testing is truly possible and convenient. Therefore, optimization of test will be studied as a separate problem in Chapter 5. The three objectives to optimize in this case are: Average Probability of Failure on Demand (PFD_{avg}), Spurious Trip Rate (STR) and Lifecycle Cost (LCC).

Figure 4.3 shows the SIS, which is composed of four subsystems. The field instrumentation comprises both the pressure and temperature sensor/transmitter subsystems, and the final control element subsystem (basically composed of a set of actuator and shutdown valve). The logic solver subsystem is a safety PLC. These subsystems are referred as PT, TT, FC and LS respectively. The optimization is focussed on the variability of redundancy and diversity of the field instruments subsystems. It is assumed that the LS subsystem meets all requirements for a SIL 3 application, and its design is not varied during the optimization. The redundancy per subsystem can be varied between one and four components, and each component can be chosen from three different technologies in order to implement diversity. These technologies are called A, B and C. They have different failure rates, diagnostic coverage and cost specifications. All relevant dependability and lifecycle cost data is presented in Table 4.1. It is important to bring to the reader's attention that the IEC 61508 device type classification (A or B), and the three different kinds of technologies available per subsystem (named A, B, C), should not be confused. The latter will be referred to as "technology", and the former as "device Type".

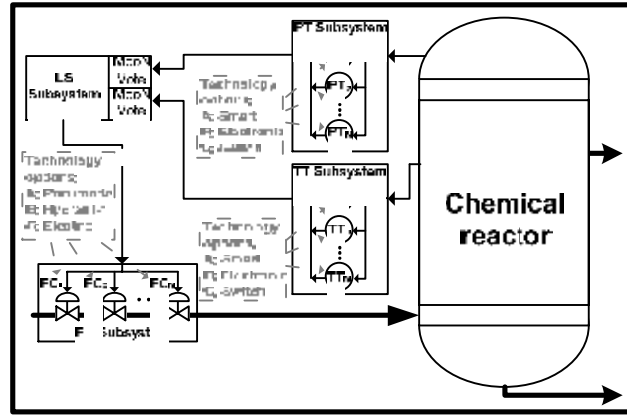


Fig. 4.3. Chemical reactor protection system

Table 4.1. Dependability and Lifecycle Cost data

Subsystem Technology	Pressure measurement/transmitter			Temperature measurement/transmitter ^b			Final control element (valve & actuator)			Logic solver
	A	B	C	A	B	C	A	B	C	
	Smart transmitter	Conventional Electronic transmitter	Switch	Smart transmitter	Conventional Electronic transmitter	Switch	Air operated ^c	Hydraulic operated ^d	Motor operated ^d	Safety PLC
ϵ^D/ϵ^D (%)	69.2/31.8	56.0/51.1	10/10	97.9/7.5	86.9/45.5	10/10	0/25	0/20	0/10	-/81.25
λ^{SD} ($\times 10^{-6}/hr$)	0.265	1.21	0.68	5.05	6.5	0.92	0	0	0	3.46 ^f
λ^{SU} ($\times 10^{-6}/hr$)	0.118 ^a	0.95	6.13	0.11 ^a	0.98	8.30	3.94	3.17	9.17	
λ^{DD} ($\times 10^{-6}/hr$)	0.048	0.97	0.41	0.026	1.57	0.76	0.84	1.09	0.79	0.026
λ^{DU} ($\times 10^{-6}/hr$)	0.103	0.93	3.70	0.322	1.88	6.84	2.51	4.35	7.11	0.006
λ^T ($\times 10^{-6}/hr$)	0.534	4.06	10.92	5.508	10.93	16.82	7.29	8.61	17.07	3.492
Type	B	A	A	B	A	A	A	A	A	B
SFF (%)	80.7	77.09	66.12	94.15	82.79	59.33	65.57	49.48	58.35	99.83
C _{purchase} (\$)	4844	3206	500	2560	1406	500	6940	6400	6200	40000

Lifecycle cost data:

Design/install/commissioning PLC= 10,320 (\$)
 Repair PLC= 8000 (\$/event)
 Maintenance PLC= 960 (\$/event)
 Test PLC= 240 (\$/event)
 Design overall instrumentation= 3,060 (\$)
 Installation/commissioning per instrument=600 (\$)
 Maintenance per instrument= 240 (\$/event)
 Test per instrument= 60 (\$/event)
 Repair cost per instrument & PLC= 480 (\$/event)
 Spares per repair= 7.5% component cost
 Loss of production= 2,000 (\$/hour)
 Start up cost= 1800 (\$)
 Catastrophic loss= 150×10^6 (\$)
 SIS life= 15 (years)
 Discount rate=0.05

Other data:

Repair time=8 (hrs)
 Shut down time= 24 (hrs)
 Test Interval= 1 (year)
 Plant risk without SIS= 8.55×10^{-3} /yr

$C_{1002}=1.0$

$C_{1003}=0.3$

$C_{1004}=0.15$

$\beta=10\%$ ^e

Notes:^aThis failure mode doesn't cause spurious trip, only quantified for SFF^bThe measurement element is a thermocouple^cIncludes the solenoid valve^dIncludes the electromechanical relay^eApply for all cases, except for β^{SD} of: PTA=5%, TTA=2%, TTB=5%^fThis figure includes $\lambda^{SD} + \lambda^{SU}$

A second optimization case that constrains the design to only variation in redundancy without diversity is also studied. In this, the level of redundancy can be varied but all the components of each redundant subsystem must be of the same technology. The objective is to contrast the case of optimization of design with diversity against the optimization without mixing components.

The following assumptions apply for the solution of the problem:

- The system operates in low-demand mode.
- Components have constant failure rate.
- Failure of each subsystem is independent from others.
- Once a component has failed it remains in that state until it is repaired.
- Testing and repair are perfect: no faults are overlooked, and the item is returned to “as new” condition or normal state after repair.
- The testing strategy is independent test per component.
- For redundancy higher than two only the CCF that affects all redundant components is considered (in the same fashion as suggested by Hauge et al., 2006a).

The different β factors for each kind of technology per subsystem shown in Table 4.1 have been estimated using the tables presented in IEC 61508 Part 6 Annex D. Since this is considered a design made in an early project stage, the only categories relevant for estimating the β factors were: complexity/design/application/maturity/experience; assessment/analysis and feedback of data; procedures/human interface; environmental control; environmental testing. Specific β factors for each failure mode (dangerous detected, dangerous undetected, safe detected and safe undetected) have been determined. However, practically the β factor in every case is the same (10%), with the few exceptions of safe detected failure mode for PT technology A which is 5%, TT technology A is 2%, and TT technology B is 5%.

For simplicity it has been assumed that appropriate measures have been put in place to eliminate coupling among different technologies, so that common cause failure events are negligible. This means that CCF is only considered between components of same technological type. Quantification of CCF caused by coupling factors between different technologies is a very complex task. In addition, remember that different CCF rates can take place between the four failure modes (dangerous detected, dangerous undetected, safe detected and safe undetected). It therefore would require a CCF model capable of handling quantification of contributions by different couplings in a way traceable to actual design and plant conditions, and a combinatorial model capable of handling them efficiently in the changing conditions of the design optimization process. This is an opportunity of research proposed as such in the Future Work section of the last chapter.

4.4. QUANTIFICATION OF DEPENDABILITY OBJECTIVES

4.4.1. Fault Tree Analysis

The two dependability objectives, PFD_{avg} and STR, are quantified based on Fault Tree Analysis. The fault trees are empowered with house events, which confer the fault trees the flexibility to accommodate the dynamic changes in redundancy and technology during the optimization process. Figure 4.4 and 4.5 show the fault tree for quantification of PFD_{avg} and STR respectively. In the fault trees of Figures 4.4 and 4.5 the house events allow switching on and off the redundant units per technology per subsystem by changing their logic value to 1 or 0.

Notice that the full fault trees are not displayed; only the full branch for technology A of the PT subsystem is shown, which is exactly the same for technologies B and C. In the same fashion, the sub-fault trees for the TT and FC subsystems would be identical to the entire PT subsystem sub-tree.

The fault trees have as basic events the independent and CCF events of each subsystem. Thus, each basic event is composed by a detected and undetected component (as in Eq. 1.19). The formulas for these basic events of both fault trees are shown below. Notice the separation between independent (normal) and common cause failures, formulated in Eqs. (4.3) and (4.4). Also observe that the β factors are modified by the factor C_{MooN} given by Eq. (4.5).

For dangerous independent failure basic events:

$$PFD_{avg} = (1 - C_{MooN} \cdot \beta^{DD}) \cdot \lambda^{DD} \cdot T_r + (1 - C_{MooN} \cdot \beta^{DU}) \cdot \lambda^{DU} \cdot \left(\frac{TI}{2} + T_r\right) \quad (4.8)$$

For dangerous common cause failure:

$$PFD_{avg} = C_{MooN} \cdot \beta^{DD} \cdot \lambda^{DD} \cdot T_r + C_{MooN} \cdot \beta^{DU} \cdot \lambda^{DU} \cdot \left(\frac{TI}{2} + T_r\right) \quad (4.9)$$

For safe independent failure basic events:

$$STR = (1 - C_{MooN} \cdot \beta^{SD}) \cdot \lambda^{SD} + (1 - C_{MooN} \cdot \beta^{SU}) \cdot \lambda^{SU} \quad (4.10)$$

For safe common cause failure:

$$STR = C_{MooN} \cdot \beta^{SD} \cdot \lambda^{SD} + C_{MooN} \cdot \beta^{SU} \cdot \lambda^{SU} \quad (4.11)$$

The fault trees are solved by quantification of MCS. The program developed for quantification of the objective function selects dynamically the cut sets relevant to the candidate solution under evaluation. It then calculates the total PFD_{avg} and the STR of this candidate solution.

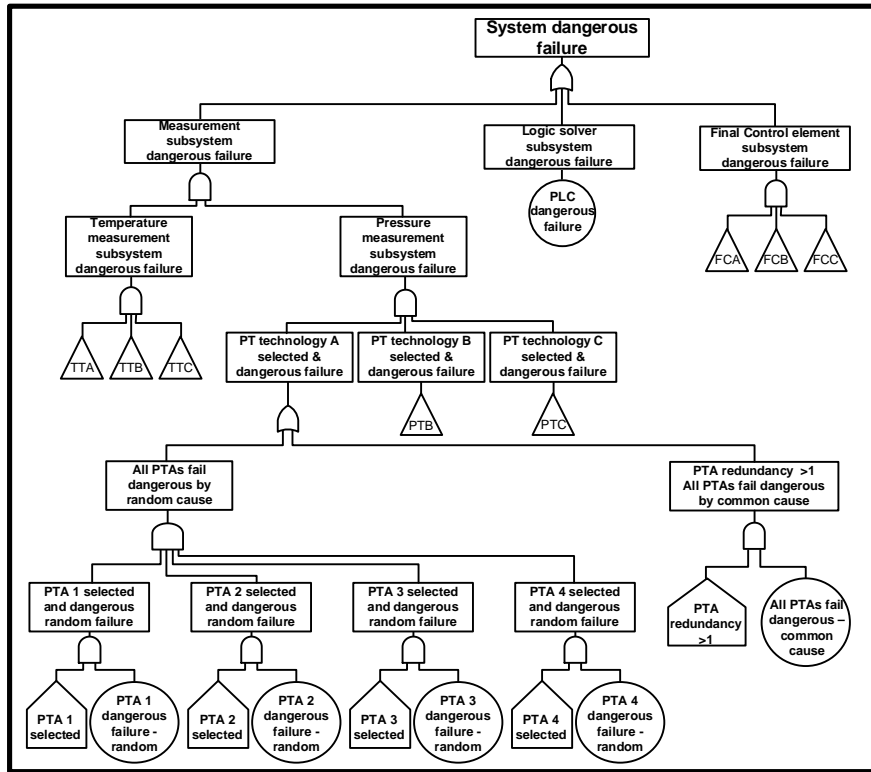


Figure 4.4. Fault tree for quantification of Probability of Failure on Demand

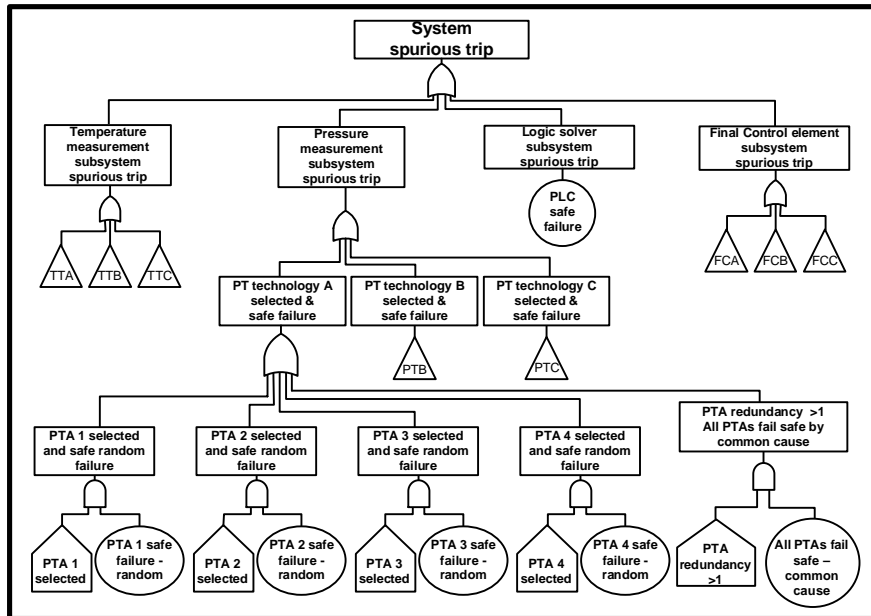


Figure 4.5. Fault tree for quantification of Spurious Trip Rate

4.4.2. Computer code for solving fault trees

As it can be seen comparing the fault trees presented here with those of Chapter 3 (Figs. 3.6 and 3.7), their size has grown considerably which presents a challenge for their quantification. A program that reduces the MCS to those applicable to the specific architecture chosen by the optimizer has been developed based on MATLAB[®]. This program is used in the last two out of the four steps to quantify the PFD_{avg} :

1. Draw the fault trees with house events representing all possible basic events.
2. Obtain the MCS of the complete tree (assuming all the basic events are included in the system configuration). They have been obtained using the program OpenFTA[©] (FSC¹ 1991).
3. Reduce the MCS to those only applicable based on the house events corresponding to the actual system configuration selected.
4. Calculate the PFD_{avg} .

The program's working principle is described by an example illustrated in Figure 4.6. This considers only the subsystem FC with a combination of two components A and two C.

1. The program is fed with a vector with of all MCS and another vector with all possible combinations by subsystem (in this case they are 34 combinations).
2. A specific combination to be evaluated is chosen (this would be determined by a specific phenotype in the GA application). For example combination number 1 is AACC.
3. Create a mask of applicable house events in order to delete irrelevant terms (those that do not belong to the chosen combination AACC) from all the MCS, and apply it to filter them out. For example, one MCS is:

$$MCS_1 = [FCB_{CCF} \cdot FCC_{CCF} \cdot FCA_1 \cdot FCA_2 \cdot FCA_3 \cdot FCA_4]$$

In this example FCB_{CCF} represents the CCF of the FC element for technology B. FCA_1 represents the independent failure of component 1 of the FC technology A. As can be seen in Figure 4.6 the terms in this MCS relevant to the combination AACC would be only FCC_{CCF} , FCA_1 and FCA_2 . Therefore the mask would permit only those terms to be passed.

The irrelevant terms are substituted by a 1.

4. Evaluate the MCS in turn.
5. Calculate the PFD_{avg} summing up all cut sets.

The combination for the three subsystems (PT, TT and FC) is determined by its corresponding chromosome (transformed into a phenotype) in the GA. The computer program for quantification of STR works in a similar fashion. Then the two obtained values PFD_{avg} and

¹ Formal Software Construction Limited

STR, together with the LLC, represent the solution of the objective function for the specific chromosome.

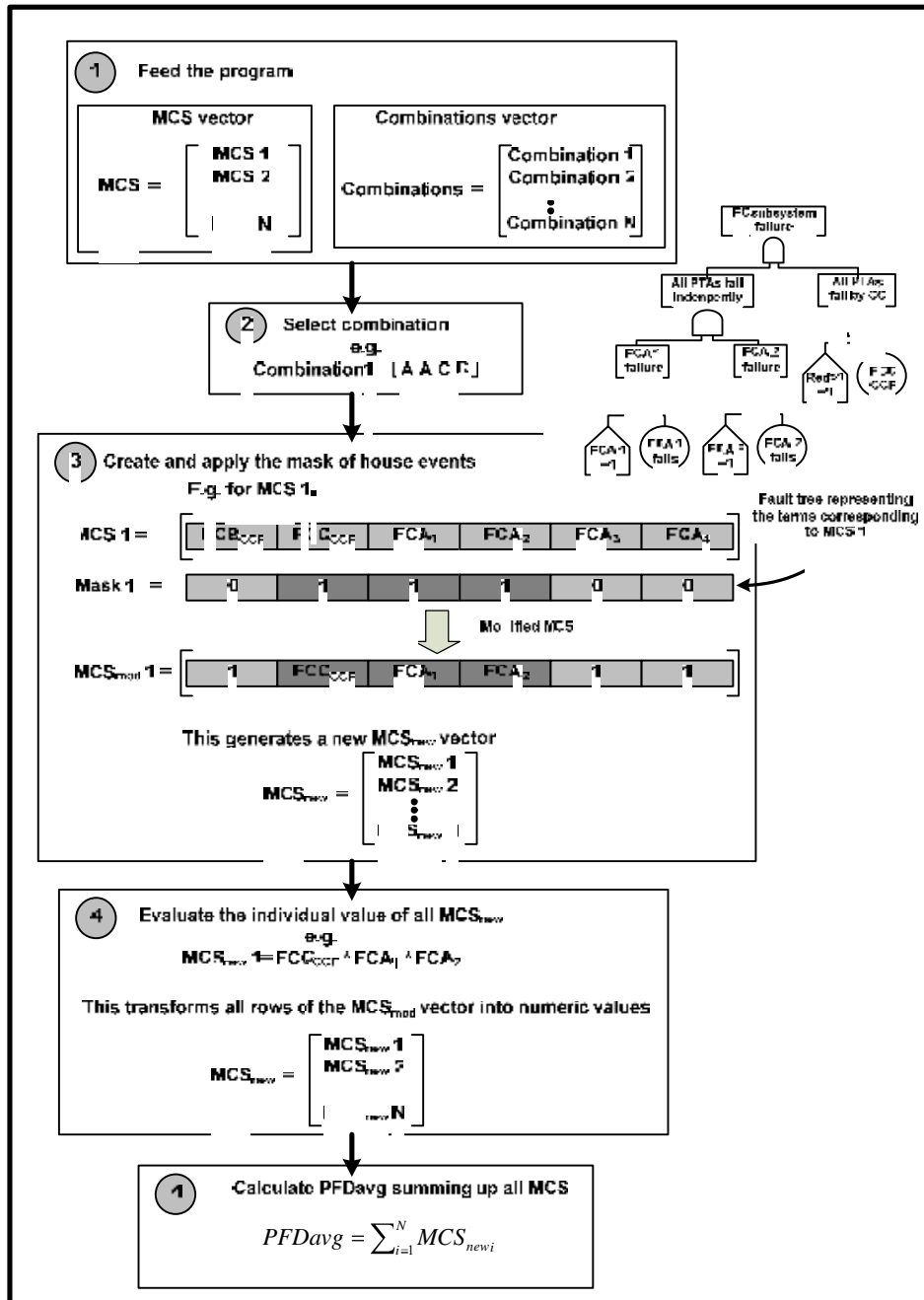


Figure 4.6. Algorithm for solving fault trees based on MCS reduction

4.5. THE LIFECYCLE COST MODEL

The LCC model has been modified (from the previous chapter) to the requirements of the new problem. It is fully included here for the sake of completeness. The overall LCC is given by:

$$LCC = C_{PROC} + (C_{OP} + C_{RISK}) \cdot PVF \quad (4.12)$$

The applied factor by present value (PVF) is determined in Eq. (3.2). The procurement cost per year is:

$$C_{PRO} = C_{design} + \sum_{N_{ij}} (C_{ij}^{purchase} + C_{ij}^{inst/comm}) N_{ij} \quad (4.13)$$

Where N_{ij} is the number of components of the j^{th} type of technology of the i^{th} subsystem. Basically, these are the decision variables of the problem, and:

C_{design} = Design cost

$C_{purchase}$ = Purchase cost

$C_{inst/comm}$ = installation, commissioning and start-up costs

The operating cost per year (Eq. (4.14)) includes preventive maintenance (C_{PM}), testing (C_T) and corrective maintenance (C_{CM}), calculated by Eqs. (4.14-4.17).

$$C_{OP} = C_{PM} + C_T + C_{CM} \quad (4.14)$$

Where:

$$C_{PM} = \sum_{N_{ij}} \frac{1}{M_{ij}} C_{ij}^{PM} \cdot N_{ij} \quad (4.15)$$

$$C_T = \sum_{N_{ij}} \frac{1}{TI_{ij}} C_{ij}^T \cdot N_{ij} \quad (4.16)$$

$$C_{CM} = \sum_{N_{ij}} f_{ij}^{CM} \cdot C_{ij}^{CM} \cdot N_{ij} \quad (4.17)$$

Where M is the maintenance frequency, TI the test interval, and f the frequency of repair. The costs C_{ij}^{PM} , C_{ij}^T and C_{ij}^{CM} are costs per event. The repair frequency is given by:

$$f_{ij}^{CM} = \lambda_{ij}^{Total} \quad (4.18)$$

The risk cost per year includes both the costs caused by safe and dangerous failures:

$$C_{RISK} = C_{STR} + C_{HAZARD} \quad (4.19)$$

The cost for spurious trip rate C_{STR} is directly proportional to the cost of production loss from each spurious shutdown C_{SD} (Eq. (4.20)). This is in turn proportional to the restoration time of the plant after a spurious shutdown (Eq. (4.21)).

$$C_{STR} = STR \cdot C_{SD} \quad (4.20)$$

$$C_{SD} = SD_{Time} \cdot SD_{Loss} \quad (4.21)$$

The cost of hazard contemplates the cost of a potential accident (Eq. (4.22)), where $F(\text{ACC}|\text{PFD}_{\text{avg}})$ is the frequency of plant failure (or accident) per year without the SIS (see Eq. 3.11).

$$C_{\text{HAZARD}} = C_{\text{ACC}} \cdot F(\text{ACC}|\text{PFD}_{\text{avg}}) \cdot \text{PFD}_{\text{avg}} \quad (4.22)$$

4.6. IMPLEMENTATION OF THE OPTIMIZATION ALGORITHM

Same as in Chapter 3, the optimization of the safety instrumented system design was based on the Fonseca & Fleming MOGA genetic algorithm (Fonseca & Fleming, 1993). The optimization program was implemented in MATLAB[®], using the MOGA toolbox (Chipperfield et al., 1994). The implementation of the optimization algorithm follows the flowchart presented in Figure 4.7.

For each single subsystem there are 34 unique combinations of redundancy and technologies (making a total of about 40×10^3 possible combinations for the design of the overall system), which were codified and included in a vector (see Section 4.4.2). The vector permits access to these combinations by their index. Notice that the decision variables are integer, which favours the use of a binary chromosome. The 21-bit binary chromosome (seven bits per subsystem, as shown in Figure 4.8), access the relevant combination in the combinations vector according to the candidate solutions being evaluated. With this information the objective function is called and calculated, which returns a vector with values of the objectives under optimization. These vectors are stored in an Objectives Matrix. After the genetic operators are applied to the Objectives Matrix, another Best Matrix collects and stores the best-ranked individuals from each generation. Once the program termination criterion is met (i.e. maximum number of generations is exhausted) this matrix is re-ranked, in order to eliminate all the dominated individuals. The resulting pool of solutions constitutes the Pareto Optimal Front.

Tuning was made for the algorithm, and the parameters that provided better results (more diversity and individuals in the Pareto front) were:

- Population: 20 individuals
- Generations: 300
- Generational gap: Incremental from 10 to 70%
- Fitness mapping: Exponential
- Selection algorithm: Stochastic Universal Sampling
- Crossover mechanism/rate: Single point at 70%
- Mutation mechanism/rate: Bit-flipping at 5%

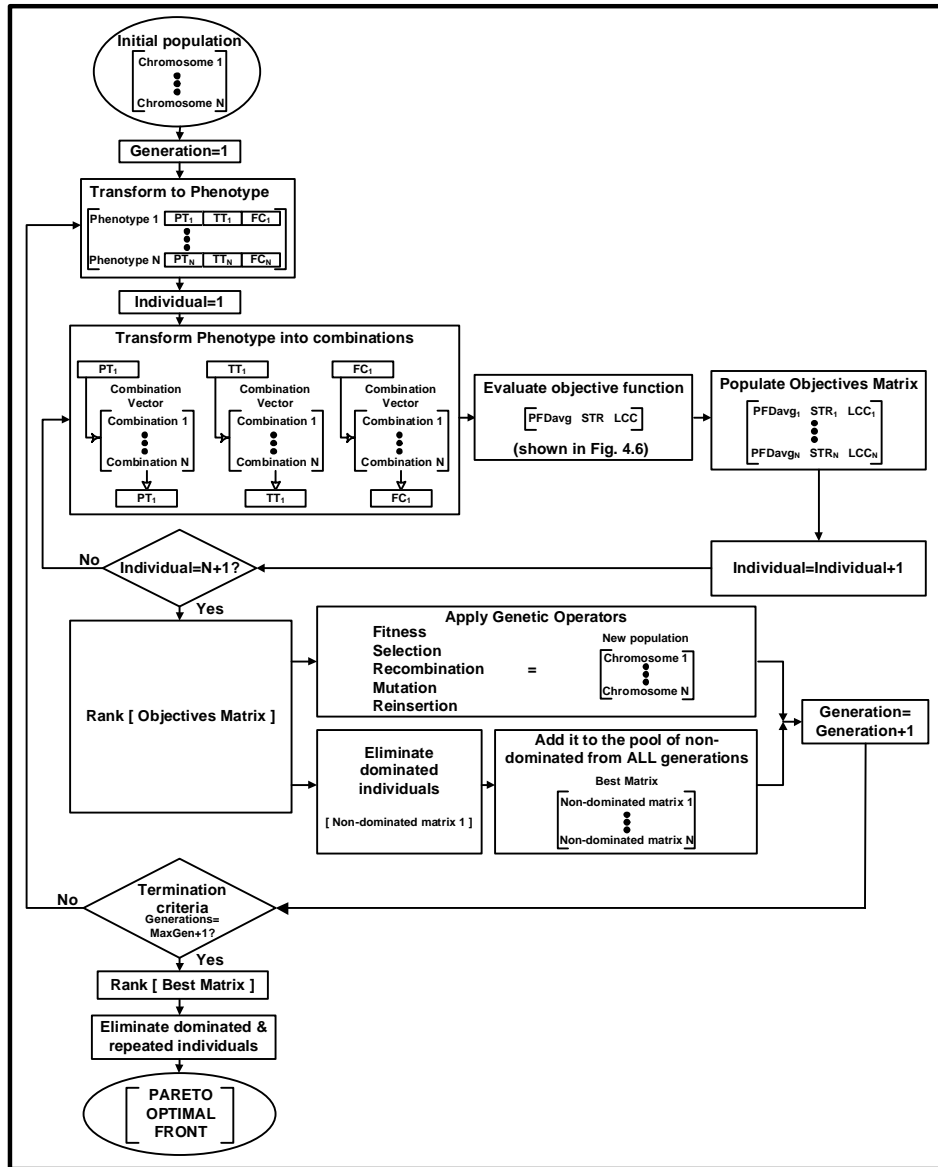


Figure 4.7. Flowchart of the implementation of the optimization algorithm

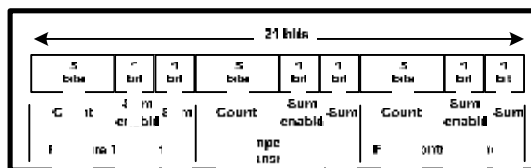


Figure 4.8. Chromosome code for the SIS design

4.7. DISCUSSION OF RESULTS

4.7.1. Architectures and their performance

Table 4.2 presents the Pareto-optimal front solutions numbered and sorted by decreasing PFD_{avg} . The table shows the corresponding decision variables (i.e. the number of elements per technology per subsystem, under the headings A, B, C) and the objectives' values of each solution. The Diversity Index values given by DI_1 and DI_2 are also presented.

Analyzing the table, it seems that some technologies are much less competent for optimal design. Specifically, observe technologies C for the two transmitter subsystems. These appear only in a very few solutions, and presumably only when it is very necessary to further improve diversity (solutions 56 to 64). In contrast, smart transmitters appear in all the optimal solutions, which indicate that they are preferred by the optimizer. Smart transmitters have significantly higher acquisition cost than the other two options (see Table 4.1). Nevertheless, they are still used in all the 64 solutions given by the optimizer. This suggests that for high dependability requirements quality (low failure rates and high diagnostic coverage) outweighs cost.

The case of temperature transmitters is particularly notable. The TT and PT subsystems are in parallel, eliminating the need for implementation of high levels of redundancy in both of them (see solutions 1 to 46). Temperature transmitters have redundancy higher than two only in solutions with very low PFD_{avg} values. It is evident that the optimizer gives preference to the PTs over the TTs, even though TTs have lower purchase costs. Notice that TTs have higher total failure rates than PTs with the same technology, particularly higher safe failure rates. This has a negative impact on the cost of losses by higher spurious trip rates. It can be said that the dominance of these costs exerts considerable influence on the optimizer so that the optimizer discards the TTs as a better choice over the PTs.

Table 4.2 presents two columns with SIL values. The first one is the SIL that the systems would achieve according to its PFD_{avg} (Table 1.1). The second value is the final SIL, which takes into account the architectural constraints (Table 1.2). Observe that in most of the solutions the final SIL is limited by the architectural constraints to a lower level than the one corresponding to their PFD_{avg} (highlighted in red). This suggests that for these solutions better levels of SFF and FT are required to make them congruent with the PFD_{avg} . This may be interpreted as that the nominal failure rates of the equipment may be overoptimistic. Nevertheless, it is important to highlight that IEC 61508 only indicates the SIL for up to a maximum Fault Tolerance of 2 (Table 1.2). This has been interpreted as a restriction on taking credit for a larger SIL with higher FT. It is worth mentioning that if this were permitted, more solutions in Table 4.2 would

have a higher final SIL. Specifically, solutions 59 and from 61 to 64 would achieve SIL 4.

Table 4.2. Optimization results with diversity index

Sol #	SUBSYSTEM									RESULTS							
	PT			TT			FC			PFD _{avg}	STR (year)	SIL PFD _{avg}	SIL Final	LCC (x10 ⁵ \$)	DI		
	A	B	C	A	B	C	A	B	C						DI ₁	DI ₂	DI ₃
1	1	0	0	1	0	0	0	1	0	1.91236952329x10 ⁻²	0.105	1	1	3.983	0.00	0.00	0.00
2	2	0	0	1	0	0	0	1	0	1.91231201575x10 ⁻²	0.107	1	1	4.080	0.08	0.17	0.00
3	3	0	0	1	0	0	0	1	0	1.91230751771x10 ⁻²	0.109	1	1	4.178	0.07	0.22	0.00
4	1	0	0	1	0	0	1	0	0	1.10477752329x10 ⁻²	0.111	1	1	2.978	0.00	0.00	0.00
5	2	0	0	1	0	0	1	0	0	1.10472001575x10 ⁻²	0.114	1	1	3.075	0.08	0.17	0.00
6	3	0	0	1	0	0	1	0	0	1.10471551771x10 ⁻²	0.116	1	1	3.173	0.07	0.22	0.00
7	1	0	0	1	0	0	0	2	0	2.23221566451x10 ⁻³	0.130	2	1	1.936	0.08	0.17	0.00
8	2	0	0	1	0	0	0	2	0	2.23164058915x10 ⁻³	0.132	2	2	2.033	0.17	0.33	0.00
9	3	0	0	1	0	0	0	2	0	2.23159560875x10 ⁻³	0.134	2	2	2.131	0.16	0.39	0.00
10	4	0	0	1	0	0	0	2	0	2.23158602014x10 ⁻³	0.137	2	2	2.228	0.15	0.42	0.00
11	1	0	0	1	0	0	1	1	0	2.37630341174x10 ⁻⁴	0.139	3	2	1.754	0.17	0.33	0.17
12	2	0	0	1	0	0	1	1	0	2.37055265814x10 ⁻⁴	0.141	3	2	1.851	0.25	0.50	0.17
13	3	0	0	1	0	0	1	1	0	2.37010285417x10 ⁻⁴	0.144	3	2	1.949	0.24	0.56	0.17
14	4	0	0	1	0	0	1	1	0	2.37000696805x10 ⁻⁴	0.146	3	2	2.047	0.23	0.58	0.17
15	1	1	0	1	0	0	1	1	0	2.36993721879x10 ⁻⁴	0.158	3	2	1.920	0.33	0.67	0.33
16	3	1	0	1	0	0	1	1	0	2.36991186720x10 ⁻⁴	0.163	3	2	2.115	0.29	0.83	0.25
17	1	0	0	1	0	0	1	2	0	5.14761014431x10 ⁻⁵	0.164	4	2	1.931	0.15	0.44	0.11
18	2	0	0	1	0	0	1	2	0	5.09010260834x10 ⁻⁵	0.166	4	2	2.028	0.23	0.61	0.11
19	3	0	0	1	0	0	1	2	0	5.08560456863x10 ⁻⁵	0.169	4	2	2.125	0.22	0.67	0.11
20	1	0	0	1	0	0	2	1	0	5.00994103623x10 ⁻⁵	0.170	4	2	1.997	0.15	0.44	0.11
21	2	0	0	1	0	0	2	1	0	4.95243350025x10 ⁻⁵	0.172	4	2	2.094	0.23	0.61	0.11
22	3	0	0	1	0	0	2	1	0	4.94793546054x10 ⁻⁵	0.175	4	2	2.192	0.22	0.67	0.11
23	4	0	0	1	0	0	2	1	0	4.94697659932x10 ⁻⁵	0.177	4	2	2.290	0.21	0.69	0.11
24	1	1	0	1	0	0	2	1	0	4.94627910677x10 ⁻⁵	0.189	4	2	2.163	0.31	0.78	0.28
25	2	1	0	1	0	0	2	1	0	4.94604398146x10 ⁻⁵	0.191	4	2	2.260	0.30	0.89	0.22
26	1	0	0	1	0	0	1	3	0	3.35589320790x10 ⁻⁵	0.193	4	2	2.149	0.13	0.50	0.08
27	1	0	0	1	0	0	2	2	0	2.98222459436x10 ⁻⁵	0.195	4	2	2.196	0.13	0.50	0.08
28	2	0	0	1	0	0	2	2	0	2.92471705838x10 ⁻⁵	0.197	4	2	2.293	0.21	0.67	0.08
29	3	0	0	1	0	0	2	2	0	2.92021901867x10 ⁻⁵	0.200	4	2	2.390	0.20	0.72	0.08
30	4	0	0	1	0	0	2	2	0	2.91926015744x10 ⁻⁵	0.202	4	2	2.488	0.19	0.75	0.08
31	1	1	0	1	0	0	2	2	0	2.91856266489x10 ⁻⁵	0.214	4	2	2.362	0.29	0.83	0.25
32	2	1	0	1	0	0	2	2	0	2.91832753958x10 ⁻⁵	0.216	4	2	2.459	0.27	0.94	0.19
33	3	1	0	1	0	0	2	2	0	2.91830914890x10 ⁻⁵	0.219	4	2	2.557	0.25	1.00	0.17
34	2	2	0	1	0	0	2	2	0	2.91830401816x10 ⁻⁵	0.233	4	2	2.616	0.25	1.00	0.17
35	1	0	0	1	0	0	2	1	1	2.78905818210x10 ⁻⁵	0.251	4	2	2.451	0.19	0.75	0.17
36	2	0	0	1	0	0	1	2	1	2.73584661064x10 ⁻⁵	0.247	4	2	2.482	0.27	0.92	0.17
37	2	0	0	1	0	0	2	1	1	2.73155064612x10 ⁻⁵	0.253	4	2	2.548	0.27	0.92	0.17
38	3	0	0	1	0	0	1	2	1	2.73134857093x10 ⁻⁵	0.249	4	2	2.579	0.26	0.97	0.17
39	4	0	0	1	0	0	1	2	1	2.73038970971x10 ⁻⁵	0.251	4	2	2.677	0.25	1.00	0.17
40	1	1	0	1	0	0	1	2	1	2.72969221716x10 ⁻⁵	0.263	4	2	2.551	0.35	1.08	0.17
41	3	0	0	1	0	0	2	1	1	2.72705260641x10 ⁻⁵	0.255	4	2	2.646	0.26	0.97	0.17
42	4	0	0	1	0	0	2	1	1	2.72609374519x10 ⁻⁵	0.257	4	2	2.744	0.25	1.00	0.17
43	1	1	0	1	0	0	2	1	1	2.72539625264x10 ⁻⁵	0.269	4	2	2.617	0.35	1.08	0.33
44	2	1	0	1	0	0	2	1	1	2.72516112733x10 ⁻⁵	0.272	4	2	2.714	0.34	1.19	0.28
45	3	1	0	1	0	0	2	1	1	2.72514273664x10 ⁻⁵	0.274	4	2	2.812	0.31	1.25	0.25
46	2	2	0	1	0	0	2	1	1	2.72513760591x10 ⁻⁵	0.289	4	2	2.871	0.31	1.25	0.25
47	2	1	0	2	0	0	2	1	1	2.72513754906x10 ⁻⁵	0.315	4	3	2.996	0.37	1.36	0.28
48	1	1	0	1	1	0	2	1	1	2.72513705522x10 ⁻⁵	0.335	4	2	3.001	0.52	1.42	0.50
49	3	1	0	2	0	0	2	1	1	2.72513568894x10 ⁻⁵	0.317	4	3	3.094	0.35	1.42	0.25
50	2	2	0	2	0	0	2	1	1	2.72513516999x10 ⁻⁵	0.332	4	3	3.153	0.35	1.42	0.25
51	2	1	0	1	1	0	2	1	1	2.72513511261x10 ⁻⁵	0.337	4	2	3.098	0.50	1.53	0.44
52	3	1	0	1	1	0	2	1	1	2.72513496067x10 ⁻⁵	0.340	4	2	3.196	0.48	1.58	0.42
53	2	2	0	1	1	0	2	1	1	2.72513491828x10 ⁻⁵	0.354	4	2	3.255	0.48	1.58	0.42
54	2	1	0	2	1	0	2	1	1	2.72513491781x10 ⁻⁵	0.381	4	2	3.380	0.48	1.64	0.39
55	3	1	0	2	1	0	2	1	1	2.72513490244x10 ⁻⁵	0.383	4	3	3.478	0.46	1.69	0.36
56	2	1	1	1	1	0	2	1	1	2.72513489941x10 ⁻⁵	0.397	4	2	3.442	0.54	1.83	0.50
57	2	2	0	1	2	0	2	1	1	2.72513489828x10 ⁻⁵	0.416	4	3	3.620	0.46	1.69	0.36
58	3	1	0	1	1	1	2	1	1	2.72513489783x10 ⁻⁵	0.420	4	3	3.648	0.53	1.92	0.47
59	3	1	0	3	1	0	2	1	1	2.72513489783x10 ⁻⁵	0.428	4	3	3.766	0.53	1.75	0.33
60	2	1	1	2	1	0	2	1	1	2.72513489625x10 ⁻⁵	0.440	4	3	3.724	0.52	1.94	0.44
61	2	1	1	3	1	0	2	1	1	2.72513489599x10 ⁻⁵	0.485	4	3	4.013	0.60	2.00	0.42
62	2	2	0	1	2	1	2	1	1	2.72513489596x10 ⁻⁵	0.497	4	3	4.072	0.65	2.00	0.42
63	1	2	1	2	2	0	2	1	1	2.72513489593x10 ⁻⁵	0.517	4	3	4.149	0.60	2.00	0.42
64	2	1	1	2	1	1	2	1	1	2.72513489590x10 ⁻⁵	0.521	4	3	4.177	0.71	2.25	0.50

The parallel coordinates graph is presented in Figure 4.9. This graph shows the optimal solutions with their optimized objectives in normalized parallel coordinates. Observe that the PFD_{avg} is in logarithmic scale, and that the SIL achieved by this figure may be different from the final SIL (further commented below). On the other hand, crossing lines confirm that the PFD_{avg} is generally in conflict with both the STR and the LCC. Nevertheless, there are some exceptions to this observation for SIL levels 1 and 2, where PFD_{avg} and LCC are sometimes harmonious (non-conflicting) objectives, which will better observed in Figure 4.11 later.

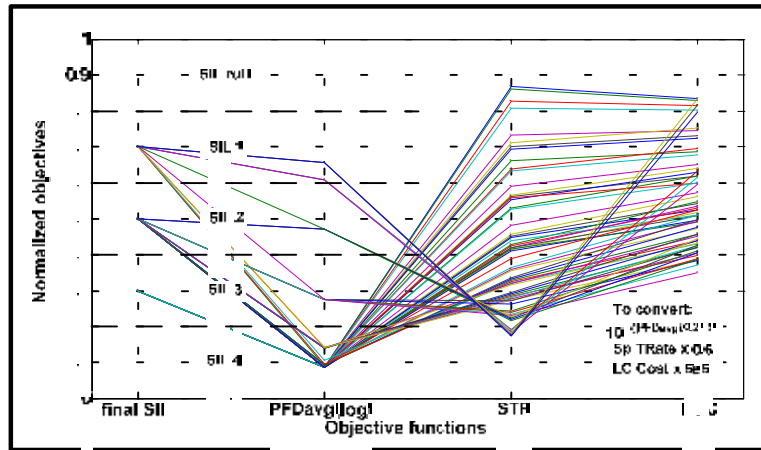


Figure 4.9. Parallel coordinates graph of the optimal set

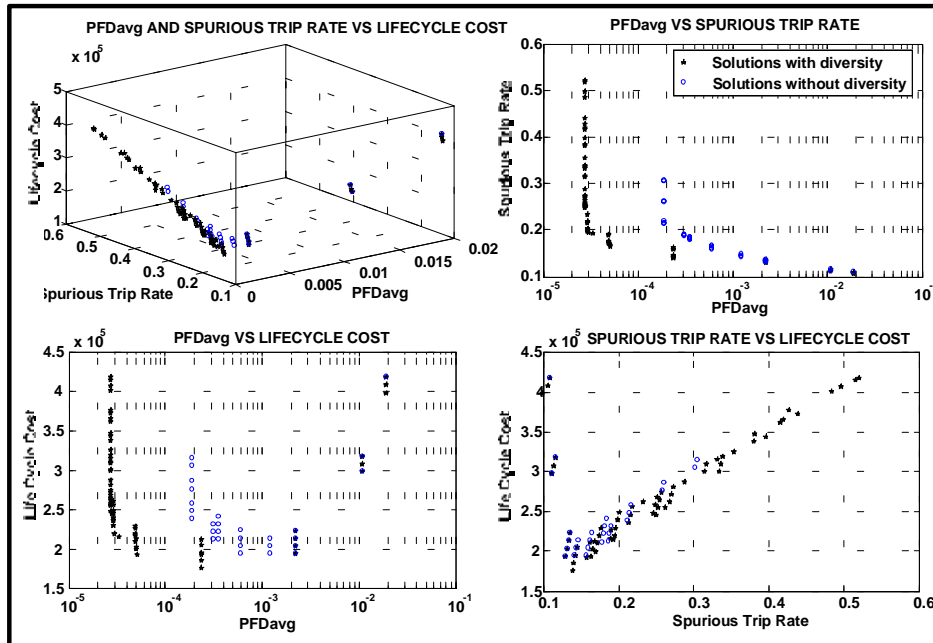


Figure 4.10. Comparison of diverse vs non-diverse solutions

Figure 4.10 serves a dual purpose. First of all, it presents the graphs for the obtained optimal solutions of Table 4.2 (optimization with diversity). Secondly, it shows over the same graphs the optimal solution obtained from the optimization without diversity, enabling the reader to compare both diverse and non-diverse approaches.

It can be noticed that discontinuities in the search space exist, presumably caused by the discrete nature of the problem. Analysing Table 4.2 it is possible to find an explanation. “Jumps” are caused when both redundancy and diversity are changed simultaneously between two consecutive solutions, creating clusters of solutions and empty spaces in the search space and the Pareto-optimal front.

4.7.2. Discussion of diversity

The graphs in Figure 4.10 illustrate the most relevant demonstration of this chapter. Comparing the optimal sets obtained with and without diversity, it can be noticed that, for solutions that are not members of both sets, the non-diverse solutions are dominated by the diverse ones. This was verified re-ranking all solutions of both sets together (see Section 3.4.1) in order to get only the overall non-dominated ones. It was found that the non-diverse set did contain some few non-dominated solutions, but they were part of the set of solutions belonging to both sets. This leads us to conclude that, in general, the optimization permitting diversity gives better solutions (i.e. better trade-offs) than the one without diversity.

Analyzing the data presented in Table 4.2 it is possible to find many examples that reinforce the hypothesis that increasing diversity reduces the PFD_{avg} . The Diversity Index of several solutions is highlighted in yellow in the last columns of the table. Among these solutions several pairs can be made for which the two DIs are consistently larger for more diverse solutions, and which show improvements in the PFD_{avg} . Three different cases have been detected:

- a) The following pairs have exactly the same level of redundancy with different level of diversity in one of the subsystem, where the solutions with higher diversity have lower PFD_{avg} : 27-35, 28-36, 29-38-41, 30-39-42, 31-40-43, 32-44, 33-45, 34-46, 47-51, 50-53, 53-56, 55-58, 55-60 and 61-64. The majority of the changes in diversity are implemented in the FC subsystem, although there are cases where the change is in the PT or TT as well.
- b) For the pairs 7-11, 8-12, 9-13 and 10-14, with same redundancy and different diversity as well, the sole increment of diversity produces the simultaneous reduction of not only PFD_{avg} but LCC as well.

- c) Finally, for the adjoining pairs 14-15, 23-24, 30-31, 39-40 and 42-43 the PFD_{avg} is reduced by reducing redundancy and increasing diversity simultaneously, which also results in lower LCC. All these pairs have in common that the only change is from having four PTs of optional type A to having one PT A and one PT B, what basically eliminates CCF. Notice, however, that the reductions in both PFD_{avg} and LCC are not significant.

From those cases it could be concluded that increasing the diversity can enhance the system's PFD_{avg} , and in some cases it may even improve the LCC when considering reduction in STR.

In order to facilitate the analysis of the convenience of using either DI_1 or DI_2 as Diversity Index, these are plotted next to the number of total components, diversity and number of different technologies used in each optimal solution in Figure 4.11. In addition, the upper part of the figure shows the objective function values of the optimal set.

Recall that the Diversity Index aims to reflect the relative level of diversity in the overall system, and this was the idea behind DI_1 (Eq. (4.6)). However, since the reduction of PFD_{avg} does not solely depend on diversity but redundancy as well, DI_2 (Eq. (4.7)) was formulated.

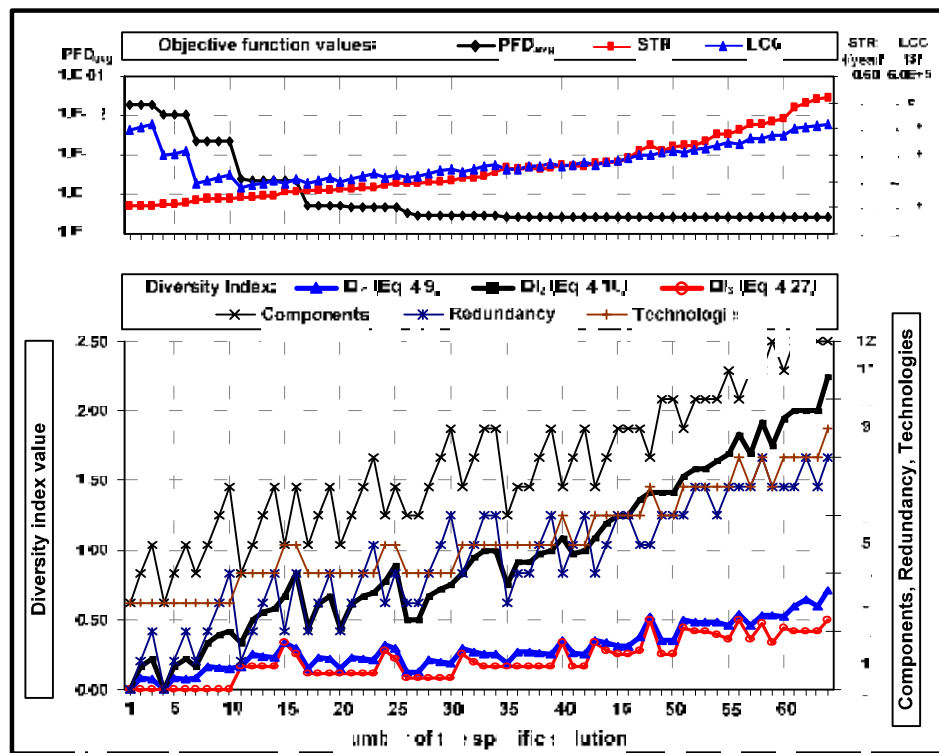


Figure 4.11. Comparison of Diversity Index vs optimized objectives

Observe the performance of DI_1 , for instance between solutions 7-10. From solution 7 to 8 the PFD_{avg} decreases as DI_1 increases, which could suggest that the increment in diversity is enhancing PFD_{avg} . However from solution 8 to 10 PFD_{avg} decreases further, but DI_1 also decreases. Thus, the improvements in PFD_{avg} are not due to diversity in this solution. Observing the plot of redundancy, it can be seen that PFD_{avg} decreased due to an increment in redundancy. This is not reflected by DI_1 . This is, however, reflected by DI_2 . Observe that this DI_2 increases steadily for solutions 7-10. This seems to be more consistent. Observe that between solutions 25-26 both redundancy and diversity decrease, and DI_2 do it as well. Therefore, DI_2 seems to reflect more consistently that the improvement in PFD_{avg} is due to both diversity and redundancy. Although DI_1 could be used for emphasizing more the contribution of diversity it can be misleading. For instance, between solutions 17-20 the diversity remains constant but DI_1 changes. This seems to be confusing. It can be seen that it is following the level of redundancy, but this indication is already given by the DI_2 . Thus, in the search of a DI that put precise emphasis solely on diversity a new formulation of diversity relative to redundancy is made in Eq. (4.23).

$$DI_3 = \frac{1}{N_s} \sum_i^{N_s} \frac{Technologies_i - 1}{Components_i} \quad (4.23)$$

The upper term is formulated as *Technologies-1* in order to rule out the possibility of a system composed of one single component being considered as a diverse arrangement. Thus, this term indicates that a subsystem can be considered diverse if it has two or more components with two or more different options. The values given by this equation are also included in Table 4.2 and plotted in Figure 4.11. The formula reflects only changes in diversity, not redundancy.

Some of the most significant changes in PFD_{avg} between two consecutive solutions of the optimal set are analyzed in Table 4.3. The major “jumps” in PFD_{avg} values between two solutions are compared against the numerical values of the DIs. From the table, notice that all these major changes in PFD_{avg} are not caused by an increment of redundancy, since its level decreases for all these cases. The number of total technologies being used actually only changes for the cases 16-17 and 25-26. However, the relative diversity only changes for 16-17.

Table 4.3. Analysis of DI change in the cases of major changes of PFD_{avg}

Solutions	Redundancy	Number of used technologies	DI_1	DI_2	DI_3
3-4	Decreases	No change	Decreases	Decreases	No change
6-7	Decreases	No change	Increases	Decreases	No change
10-11	Decreases	Increases	Increases	Decreases	Increases
16-17	Decreases	Decreases	Decreases	Decreases	Decreases
25-26	Decreases	Decreases	Decreases	Decreases	Decreases
34-35	Decreases	No change	Decreases	Decreases	No change

Observe that in solutions 1-9 there is actually no diversity, which is truly expressed by DI_3 . Both DI_1 and DI_3 show correctly an increment of diversity between solutions 10-11. However, DI_1 also shows a change in diversity for the case 6-7, which is not possible since there is not actual diversity. Therefore, DI_3 is more precise. It is notable that the major changes in PFD_{avg} indicated in the table are not caused by an increment of diversity nor redundancy. This means that the decrements of PFD_{avg} are caused by a change in the technologies being selected in the subsystems, which can be confirmed analyzing Table 4.2.

We can conclude that the Diversity Index that reflects more faithfully changes in only diversity is DI_3 . If changes in both redundancy and diversity must be reflected, DI_2 is a better option. They could actually be used together to guide better the analysis.

The Diversity Indexes DI_2 and DI_3 may still need improvement. As indicated by the cases in Table 4.3, sometimes the PFD_{avg} can be reduced when the DI_3 actually decreases. As previously commented, this happens when the choice of technologies is changed. However, the formulation of the DI is still useful and facilitates system analysis. For example, Figure 4.11 shows that there is a general trend in which as the PFD_{avg} decreases the DI tends to raise. This observation is possible due to the plotting of the DI. This is also true for all the three DI formulations. In addition, consider the analysis made at the beginning of these sections regarding couples of solutions where increasing the diversity index decreases PFD_{avg} . These relationships are valid for all the three formulations. This means that when the number of components remained constant and only the level diversity changed, this change caused a reduction of PFD_{avg} . This situation was correctly reflected by any of the DIs formulated here.

Going back to Figure 4.11, and focussing on DI_3 as Diversity Index, observe that the graph shows a general trend where the larger the DI the lower PFD_{avg} and the higher STR and LCC. The general trend leads us to think that the higher levels of diversity benefit the system safety integrity, making possible to achieve a design with lower levels of PFD_{avg} .

The relationship between PFD_{avg} and LCC is, however, harmonious for solutions 1 to 10 (see the corresponding column in Table 4.2 as well). This group of solutions has a $PFD_{avg} > 1 \times 10^{-3}$ (i.e. SIL 2 or 1). These solutions also have lower levels of redundancy than the rest. Observing DI_3 , it can be seen that there is no diversity in the subsystems ($DI_3=0$). Therefore, reductions in PFD_{avg} in these solutions cannot be attributed to diversity. The results obtained in the previous chapter showed a consistent conflictive relationship between PFD_{avg} and LCC (see Fig. 3.14). However, the optimal solutions in that chapter had all $PFD_{avg} < 1 \times 10^{-4}$, same as solutions 11 to 64 in this chapter. This suggests that for optimal solutions (with parallel architectures) where

$PFD_{avg} > 1 \times 10^{-3}$ the costs associated with dangerous failures (cost of hazards) dominate over the cost associated to safe failure (loss of production by spurious trips). Thus, reductions in PFD_{avg} also diminish LCC, becoming harmonious objectives. This relationship is inverted when $PFD_{avg} < 1 \times 10^{-3}$ because further reductions of PFD_{avg} are associated with increments of STR that now dominate the LCC.

A word of caution is necessary about the search of lower levels of PFD_{avg} . Albeit the general increment of the diversity index generally produces decreasing PFD_{avg} , the overall benefit obtained must be carefully considered. Figure 4.10 shows that in some cases some marginal improvements in PFD_{avg} can incur in very large increments in LCC and STR. Due to this situation, solutions 40 to 64 may not be useful, unless very specific requirements were being sought (e.g. either SIL 3 by architectural constraints or to achieve a specific PFD_{avg} value), such as for example the ones adopted in the nuclear energy field.

The large benefits obtained by introducing diverse redundancy are clearly consequences of CCF reduction. This is because CCF has a big impact on both dependability measures, PFD_{avg} and STR. This can be quickly identified going back to Figure 4.2. The CCF basic event dominates a redundant arrangement failure. Frequently, CCF basic events are first-order cut sets of the entire system fault tree. Therefore, their prevalent dominance is evident. The influence of CCF and its importance in the fault tree hierarchy is further analyzed through an example in Section 5.5.2.

4.8. CONCLUDING REMARKS

This chapter put special emphasis on the study of the impact of introducing diversity into the redundant subsystems upon the SIS design optimization. Looking for a practical approach, the optimization methodology has been applied to a safety function for a chemical reactor. The level of modelling detail and the assessment of functional system safety necessary to meet IEC61508 provisions are analysed, and further applied to the optimization process.

The use of redundancy to enhance the design of Safety Instrumented Systems introduces the issue of Common Cause Failure. The effects of CCF can be so substantial that it may even dominate dependability calculations, such as Probability of Failure on Demand. This has a direct negative impact on the Safety Integrity Level achieved by the system. The results presented here showed that, in general, the design optimization with diversity gives better results than the one without diversity. The results with diversity generally dominated the ones without diversity, which means that the former achieve better trade-offs between the optimized objectives.

On the other hand, the set of optimal solutions obtained showed that introducing diversity as a defence measure against CCF has a positive impact on the functional safety of the safety system, which in turn improves the overall plant safety. Specifically, it has been seen that sometimes the sole increment of diversity, even without altering system redundancy, reduces the overall PFD_{avg} . However, in contrast, this gain in safety integrity usually has an increment in some costs and the Spurious Trip Rate, which may have a negative impact in the overall system's Lifecycle Cost and in the trust given to the SIS performance. This would be justified only by the specific safety requirements of the plant.

A first attempt for quantifying the level of diversity in a SIS with a Diversity Index was made here. One formulation was made for reflecting just the change in diversity level, and a second one for reflecting change in diversity and redundancy. It has been shown that there exists a general trend to reduce PFD_{avg} at the time diversity increases. This in a few cases even reduced the system LCC. On the other hand, when the level of redundancy remains constant and the level of diversity changed, the Diversity Indexes formulated here were consistent with the improvements of PFD_{avg} . The Diversity Indexes formulated here may require further research to be refined. However, at this stage, they are a good foundation for quantification of the benefits driven by using diverse redundancy, which is a novel contribution of this work.

CHAPTER 5

Modelling and optimization of proof testing policies

This chapter has a double objective: firstly, it introduces a new development for modelling, the time-dependent Probability of Failure on Demand for parallel architectures. Secondly, it presents the integration of this model into the multi-objective optimization of proof testing policies for Safety Instrumented Systems. The article also presents an in-depth review of proof testing optimization and the current standardization and practice for SIS, specifically in the process industry. With this, the modelling and optimization are tailored for a practical approach.

The model is based on the mean test cycle, which includes the different evaluation intervals that a module goes through periodically during its time in service: test, repair and time between tests. The model is aimed at evaluating explicitly the effects of different test frequencies and strategies (i.e. simultaneous, sequential and staggered). It includes quantification of both detected and undetected failures, and puts special emphasis on the quantification of the contribution of the common cause failure on the system probability of failure on demand as an additional component. Its effectiveness is demonstrated with an application case, analyzing the sensitivity of the system dependability to diagnostic coverage and common cause failure. Subsequently, the chapter presents the multi-objective optimization of proof testing policies with Genetic Algorithms, using this model for quantification of PFD_{avg} as one of the objectives. The other two objectives are the system STR and LCC. This allows balancing of the most important aspects of safety system implementation. The overall methodology is illustrated through a practical application case of a protective system against over temperature and pressure of a chemical reactor.

5.1. OVERVIEW OF TESTING MODELLING AND OPTIMIZATION

5.1.1 Testing modelling

Proof testing is a periodic activity that mainly verifies that the specified Safety Integrity Level of the safety system is met and kept during the system lifecycle. Its more direct objective is to detect dangerous unrevealed failures. It has therefore a fundamental importance for SIS. As a consequence, periodic proof testing can contribute to achieving and improving the SIL of the system without making modifications to the safety system design. However, testing can also convey some collateral adverse effects since it is not a totally innocuous activity. It is therefore

necessary to find an optimal planning policy for the testing activity throughout the overall operational life of the system, so that a good balance between its benefits and its direct and indirect costs is achieved.

The PFD, as it was seen in Chapter 1, can be modelled using classical methods such as Fault Tree Analysis, Markov Analysis, Reliability Block Diagrams, etc. Perhaps the only method that has the flexibility to include quantification for the changing test strategy could be Markov Analysis, which is capable of accommodating failure and repair processes. However, it is well known that its complexity grows exponentially with the number of the system nodes and states.

The time dependent PFD algorithm has been developed with the specific aim to evaluate the impact that different test frequencies and strategies have on the PFD_{avg} . It also intends to include explicitly the Common Cause Failure rate of the system components (using the β factor model) and their diagnostic coverage. It is primarily based on the previous work developed by Martorell et al. (1988, 1995), Cepin & Mavko (1997) and Cepin (2002), which was intended for surveillance requirements of Nuclear Power Plants (NPP). The model addresses the level of detail required by IEC 61508 for the quantification of PFD_{avg} . The model aims to be able to accommodate changing conditions on the test strategy of the system components, with the objective of being suitable for solution of multi-objective optimization problems. Therefore, it requires a treatment that at the same time includes the modelling detail required for a real SIS application, and it does not present the disadvantage of excessive complexity growth.

An early analysis for finding the optimal Test Interval (TI) of safety related equipment of NPPs was made by Martorell et al. (1988). It established the average test cycle and quantified unavailability as the contribution of failure between tests, test time and repair time, obtaining simplified analytical equations for the sequential and staggered test strategies of a two-component parallel system. Subsequently, Uryas'ev & Vallerga (1993) proposed an approach for single objective optimization of TIs of standby safety systems. They identified four states of the instantaneous unavailability of tested components. They estimated unavailability applying probabilistic analysis tools: considered each component as a four-state Markov Chain and used fault tree's cut set quantification for calculation of the time-dependent unavailability at the system level. Unavailability minimization was solved by a numerical technique (gradient nonlinear programming method). Schofield (1993) approached the optimization of proof TIs using Fault Tree Analysis. This minimizes a function of the total time a component is under test and its Fussell-Vesely important measure subject to a top event frequency constraint (in a Lagrange multiplier). It requires solution of a set of non-linear simultaneous equations, solved by numerical methods. Vaurio (1994) developed analytical expressions for quantification of

average unavailability for sequential and uniform staggered testing (uniformly distributed over the Test Interval time) for k -out-of- n systems with up to four components. It included quantification of CCF by analyzing the convenience of several multi-parameter models. In a subsequent work, Vaurio (1995a) carried out a study for optimization of test and maintenance intervals at component level (for series systems) and plant level (using fault trees' minimal cut sets) by minimization of cost subject to risk constraints. Martorell et al. (1995) approached the interaction of surveillance TI requirements and operating restrictions (i.e. Allowed Outage Time) in NPPs. They identified the unavailability components of the periodic test cycle and developed analytical expressions for each contribution based on probabilistic methods. The concept of critical and non-critical safety unavailability separate contributions was coined by Hokstad et al. (1995), who developed a model for their quantification. It included unavailability by test in the non-critical contributions given that it is scheduled (i.e. previously known). Their model accounts for both manual and automatic (diagnostic) tests, failures introduced during testing and the probability of imperfect test. This permitted the quantitative comparison of testing schemes applied to single fire and gas detectors.

Cepin & Mavko (1997) introduced an unavailability time dependent probabilistic model taking into account the different contributions during the test cycle, and adaptable to several test strategies. The model was later applied in Cepin (2002) for optimization of safety equipment outage scheduling due to testing and maintenance by minimization of risk with simulated annealing. Bukowski (2001) presented a development that permitted inclusion of periodic inspections and repairs into Markov models of safety-critical systems. Vaurio (2003) developed analytical equations for including CCF (quantified by the General Multiple Failure Rate model) in unavailability fault trees for standby safety systems, and addressed the effects of test intervals and test staggering. Lapa et al. (2003) presented a novel approach to include non-periodic tests (within an established period) adapted to optimization constrained problems, the constraints adapted to the season-changing demands of system availability (e.g. of an emergency diesel generator). The problem studied considered a single component. Curtois & Delsarte (2006) obtained an analytic expression of the optimum simultaneous inspection interval for m out of a total of n redundant components (where $m < n$, and assuming perfect inspection), minimizing a function (by direct derivation) that comprises the probability of all the n components failing between tests plus the probability of $n-m$ components failing during the test of m components. It thus finds an optimum test interval minimizing system unavailability caused only by internal failures. An expression for staggered inspection was considered as well. They found that there exists a single optimum value for the test interval of the m components.

5.1.2. Testing optimization with Genetic Algorithms

An account of optimization of system RAMS with Genetic Algorithms was given in Section 2.3. From the point of view of testing optimization the main developments in the field are the following: The first application was made by Muñoz et al. (1997) in optimization of TIs of a motor-driven pump with a single objective GA. Pattison & Andrews (1999) included maintenance TIs as one of several decision variables for safety system design optimization, incorporating the use of binary decision diagrams for quantification of dependability of the High Integrity Protection System. This methodology was later applied in Andrews & Bartlett (2003) to a water deluge system, under maintenance and spurious trip rate costs constraints. Booth studies were later repeated for multi objective optimization in Borisevic & Bartlett (2007a, b) and Riauke & Bartlett (2008).

Martorell et al. (2000) refined and applied the unavailability model developed in Martorell et al. (1994) for optimization of TIs for the High Pressure Injection System (HPIS) of a NPP. The optimization was made considering risk (unavailability) and cost. Marseguerra & Zio (2000) implemented a GA in combination with Monte Carlo Simulation for optimization of maintenance and repair policies. Giuggioli-Busacca et al. (2001) introduced the application of multi-objective GAs to safety systems, with an application to optimization of TIs for a HPIS, based on mean availability, cost and risk (exposure time). The cost function included surveillance and maintenance plus the cost of an accident (plant damage). Martorell et al. (2002) combined the optimization of TIs, preventive maintenance and allowed outage times of a NPP, so the technical specifications and maintenance requirements were addressed integrally. Vinod et al. (2004) applied a GA to optimization of in-service inspection (which included functional test). Marseguerra et al. (2004a) included the analysis of uncertainty in the parameters in optimization of TIs. The topic has been later addressed by Martorell et al. (2008)

Martorell et al. (2004) reformulated its approach as a general multi-objective RAMS+C optimization, and refined it in (2005a). This formalized the method for assisting the decision-making process in NPPs with a strategy based on: (1) technical specifications and maintenance, (2) RAMS plus cost, and (3) goals included as constraints. Martorell et al. (2005b) explored the multi-objective optimization of the plant surveillance requirements simultaneously optimizing the TIs and the test planning (strategy), based on time-dependent modelling. Martorell et al. (2006) treated the HPIS system optimization with a double nested loop. An external loop optimizes the TIs with a constrained exhaustive search, while the internal loop (a MO GA) optimizes the test staggering.

In the latest developments, Zio & Podofillini (2007b) incorporate importance measures to the GA multi-objective optimization. Subsequently, Podofillini & Zio (2008) compared the performance of the optimization Birbaum and Risk Achievement Worth important measures against the original proposition with Fussell-Vesely. Rao et al. (2007) addressed the problem of TIs optimization with uncertain parameters using a hybrid fuzzy-genetic approach (the uncertainty handled by the fuzzy sets).

This chapter deals specifically with testing optimization. The interested reader can see Martorell et al. (2007, 2008b) for a deeper treatment of maintenance (including test) modelling and optimization and their application with GAs. Also Villanueva et al. (2008) gives a survey from the point of view of uncertainty in this type of applications.

5.1.3. Consideration of testing adverse effects

As mentioned above, the testing activity also conveys some adverse effects. Obviously, the most direct one is the incurred test down time. Nevertheless, other adverse effects may be also significant, one of the most relevant being an increment in spurious trips, mostly attributable to human error. Several researchers have approached the investigation into human error effects and imperfect procedures in test. Perhaps the first contribution is Apostolakis & Bansal (1977), who considered the increment in system unavailability by probabilities of error of omission and errors of commission. Subsequently, McWilliams & Martz (1980) incorporated the effects of two types of human errors (failure to detect a fault and leaving a components in bad state after test) in the determination of the optimum test interval at component level minimizing cost or maximizing availability. This concept was used by Lee et al. (1990) in an analysis of optimal test interval of some k -out-of- n subsystems of a reactor protection system. Two types of operator errors and the probability of test-caused failure were considered. These are modelled as constant unavailability due to human error and added to the time-dependent unavailability quantification.

Kim et al. (1992, 1994) made a thorough analysis of (single-component) test adverse effects over plant risk (availability), and introduced the concept of test-caused risk transients that lead or require a plant (reactor) trip. This can be attributed to human error, equipment failure or procedure inadequacy. This is the basis of the concept applied in this thesis. This impact on plant risk was also contemplated by Cepin et al. (1994) (together by test-caused degradation) in the search of the best test interval and strategy of an actuation system. Vaurio (1995a) contemplated errors of omission (*failure to return a component to service after test*) in the optimization of test and maintenance intervals inserting them as basic events (in a similar fashion as CCF) in the logic models. Hokstad et al. (1995) also introduced a probability term

encompassing test-introduced failures and imperfect test in their reliability models for fire and gas detectors. In a similar fashion, Bukowski (2001) included imperfect inspection and repair into Markov models of safety-critical systems. Zhao et al. (2007) approached the evaluation and optimization of reliability at component level, considering imperfect inspections at non-constant intervals. Lundteigen & Rausand (2008b) presented the first monothematic study about system spurious activation of SIS, where they discussed human error during test as an additional factor in the increment of spurious activations.

5.2. PROOF TESTING PRACTICES IN THE PROCESS INDUSTRY

This chapter intends to approach the optimization of SIS testing from a practical approach. Given that SIS are mainly relevant to the process industry some of the most relevant standard and design codes have been consulted in order to find out what is the current standard and best practice in the industry. The definition of proof test given by IEC 61508-4 is a “*periodic test performed to detect failures in a safety-related systems so that the system can be restored to an “as new” condition or as close as practical to this condition*”. The need of routine maintenance action in order to detect unrevealed failures is clearly established by the standard, and proof test is one of these activities. Therefore, it has an important role in the achievement of safety integrity. According to IEC 61508-2, the frequency of proof test will be dependent upon the target failure measure associated with the SIL, the architecture, the automatic diagnostic coverage and the expected demand rate. IEC 61508-6 provides tables for determination of the PFD_{avg} for subsystems tested specifically at 6 months, 1, 2 and 10 years intervals. No more specific guidance is provided. Notice that IEC 61508-7 defines “functional testing” as an activity “*to reveal failures during the specification and design phases.*” Therefore, proof and functional testing are not the same. In addition, IEC 61511 states that the frequency of proof test “*shall be decided using the PFD_{avg} calculation*”. It remarks that different parts of the SIS may require different test frequencies.

The Norwegian Oil Industry Association (2004) standard OLF 070 requires the proof test to confirm correct operation for the entire SIS loop, including sensors, logic solver and final control elements. It provides a list of aspects to be included in the proof tests, summarized in complete system functionality. The document establishes that the test should preferably test the entire safety loop at once (integral test). This same requirement was laid out by ISA 84.01-1996 (ISA, 1996) (the first standard to formally address SIS specifically for the process industry), as it required the final element actuation to be tested in response to sensor inputs. However, if this was not possible, partial test (by subsystem) would be accepted under certain conditions. This standard considered the periodically scheduled functional testing as part of the SIS regular

maintenance. It established the necessity of providing test facilities as an integral part of the SIS. According to ISA 84.01-1996, the functional Test Interval (TI) should be selected to achieve the system SIL.

British Petroleum (1994) RP 32-6 advocates for the assessment of complete loops, giving however consent to use testing on a sectional basis (i.e. sensor, logic solver and final elements on separate intervals) when full loop testing is not practicable. Just a loose guidance is given regarding testing frequency. The document provides a list of ranges of typical test intervals (mainly for field instrumentation), all in months, typically ranging from three to 24 months (although some functions may only need to be tested between 12 and 36 or 60 months). However, it establishes that for the most critical systems the TIs must be determined based on a reliability and risk analysis. Another design code, Exxon's International Practice IP 15-7-2 (Exxon, 1999), only makes a requirement for the provision of testing facilities for protective systems which proof test interval is larger than the process continuous run length (for instance bypass valves in order to enable protective valves to be tested online).

American Petroleum Institute (2001) API RP-14C is the more prescriptive standard. It establishes the necessity of carrying out *performance testing* for confirmation of the ability of the systems *to perform the design safety function*. It establishes as a requirement the execution of the operational test at least once per year. However, it permits usage of an alternative larger TI as long as it ensures the same or higher reliability. Some guidance is given, advising monthly test of pneumatic sensors, and quarterly for electronic transmitters. This standard indicates intervals between 3 and 24 months and up to 36 or 60 for some non-critical equipment.

It can be noticed that there is no a standard, clear, generic, thoroughly developed clear guidance about the test frequency, scope, and procedures for the process industry (comparing for example with the nuclear industry, e.g. NUREG¹/CR-6141 (Samantha et al., 1994). This lack of standardization was implicitly recognized by the British Health and Safety Executive (HSE), when commissioning a study about actual proof testing practices in the chemical industry (HSE, 2002). The study confirmed the “*existence of conflict between the need for realistic proof testing and the need to minimise downtime*” (i.e. system unavailability). It establishes that the purpose of the proof testing is the detection of unrevealed fail-to-danger faults at the time of testing (different from automatic diagnostics that intend to reveal faults between tests). It also requires including all the subsystems (sensors, logic solvers and final elements) in the proof test. Although the study describes the end-to-end testing as the ideal practice, it acknowledges the

¹ A series of standard issued by the U.S. Nuclear Regulatory Commission

need to resort to partial testing as a necessary practice under certain conditions, since end-to-end testing is not always practicable. This document gives a definition for partial testing, which is considered either *“the testing of system components at different times and frequencies or the testing of sub-sets of functions of single components”*. As other documents, the guidance provided about the frequency of tests is quite loose.

In summary, there exists a trend to use the terms “functional test” and “proof test” as the same thing. However, they are not completely the same. We consider the IEC 61508 definition of proof test as the more convenient, and thus it is the one we use in this work. The objective of proof test is to detect unrevealed failures (especially dangerous ones), and it should include both inherent and functional integrity. Some standards make requirements for the inclusion of the necessary facilities so online testing is possible where this is relevant (ISA S84.01-1996, Exxon IP-15-7-2). Proof testing is required to confirm correct operation for the entire SIS loop, including sensors, logic solver and final control elements. It should be an integral end-to-end test (OLF 070, ISA S84.01, BP² RP 32-6). However, when this is not practicable partial test is widely accepted (ISA S84.01, BP RP 32-6, HSE 2002). Proof test frequency should be specified according to the required system safety integrity and its design (IEC 61508, IEC 61511, BP RP 32-6, ISA S84.01). There is no more specific guidance or constraints about how to set the test frequency. Only a few numerical examples were found (IEC 61508, BP RP 32-6, API³ RP 14C). It can be concluded that during optimization of test frequency and strategy, partial test seems to be the most convenient for manipulation rather than integral test. This is feasible since redundancies in the SIS are usually at the subsystem level rather than at the system level.

5.3. TESTING BASIC CONCEPTS

The proof test requirements of a SIS are addressed by the test policy. A proof testing policy includes the type of test, the Test Interval (TI, which determines the frequency), and the Test Strategy (TS). These two last items, denominated together TI+TS, can be manipulated to find an optimal solution (Martorell et al., 2005b; Martorell et al., 2006). They are defined under the framework of the mean test cycle, which is described below.

5.3.1. Mean test cycle

Proof testing is a periodically executed activity. The mean test cycle includes all the events between two consecutive tests of a component (Martorell et al., 1995). The component goes through several states along the cycle: Testing, repair and standby (with respect to test) or time

² British Petroleum

³ American Petroleum Institute

between tests. There is additionally a previous time before the first test of the component which is necessary to consider for quantification of unavailability. Figure 5.1 illustrates the description of the mean test cycle. Each of the states described in the figure has a contribution to the component unavailability. The *time to first test* T_p is the time that elapses between the first system start up ($t=0$) up to the time of the first test event of the component. The *test time* and *repair time* have a T_t and T_r duration time respectively. In this work they are assumed to be fixed mean values. The time between two consecutive tests of the same component, called *standby between tests*, is the time elapsed between the last test-repair event and the next test, and it is equivalent to $TI - T_t - T_r$.

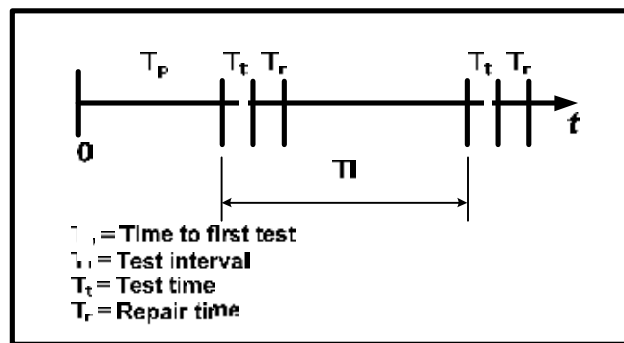


Figure 5.1. Mean test cycle

5.3.2. Test Strategies

The Test Strategy (TS) establishes how the tests of the redundant components are scheduled with respect to one another. Proof testing can be implemented using several different strategies. A general classification of TS is listed below, where T_{p_i} is the time of first test of the i^{th} component, T_t the duration time of the test, and TI the test interval.

- Simultaneous test. The N redundant components are tested at the same time, i.e. $T_{p_1} = T_{p_2} = \dots = T_{p_N}$. This implies that there are N crews available to test the N components. This is a strategy not suitable for safety systems that must remain in service permanently, since the system is made unavailable during the test.
- Sequential test. The N redundant components are tested consecutively one after another. Immediately after one component is tested and restored to work the next one is tested and so on until finishing with all the components of the subsystem. For simplification, it can be said that the only difference between the test events of two components is the time the component is under test T_t , i.e. $T_{p_2} = T_{p_1} + T_t$, ... $T_{p_N} = T_{p_{N-1}} + T_t$. The components are usually tested at the beginning or end of the test interval TI (Cepin, 1995).
- Staggered test. The N redundant components are tested with a difference of time of TI/N

between one another. This is the most common staggering strategy. This thesis explores, however, a staggered strategy where the components are scheduled for a test with difference in time larger than T_t , but not necessarily uniformly distributed as TI/N . In this thesis the former is called non-uniform staggered test and the latter uniform staggered test.

- Independent test. The relative test time of the N components are tested without any specific schedule, with a random difference of time between one another.

5.4. PFD TIME DEPENDENT MODEL

5.4.1. The PFD(t) baseline model

The PFD is the system safety unavailability, thus it requires a system unavailability model. A preliminary baseline model has been developed using as departure point the model presented by Cepin & Mavko (1997). The mean test cycle is repeated periodically with a frequency $1/TI$. This situation can be seen as a periodic “reset” of system unavailability. It is thus convenient to adopt a time dependent variable that, based on this situation, allows “re-initialization” of the time scale instead of handling a continuously-increasing value of t . Consider the definition of a variable w that will replace the usage of t in the time dependent modelling:

$$w = (t - T_p) \bmod TI \quad (5.1)$$

Where **mod** is the modulo operation (dividing remainder), which permits to reset w every time a Test Interval is completed and a new test must be performed.

The contributions that each phase of the mean test cycle has to the unavailability are formulated next:

a) Standby before first test. This is the time elapsed between $t=0$ and T_p . The unavailability during this phase is simply the temporal contribution due to the component unreliability plus a constant q . The q parameter represents a constant that depending of the application can include different contributions (as analyzed by Vaurio (1995a)), for instance the unavailability caused by detected failures (which will be explored later).

$$PFD(t) = q + 1 - e^{-\lambda t} \quad \text{for } t - T_p \leq 0 \quad (5.2)$$

b) Testing. During testing the component is taken out of service so it can be intervened. It is assumed that it is unavailable during the entire test time.

$$PFD(t) = 1 \quad \text{for } \{t - T_p > 0 \ \& \ 0 < w \leq T_t\} \quad (5.3)$$

c) Repair. This contribution to the PFD is due to the possibility of the component being found failed during test, and thus it must be repaired (assuming it is repaired immediately after test). Therefore, the unavailability is composed of two terms: one where the component is found failed at the moment of test (thus under repair), the other where the component is not failed (i.e. working) subject to the probability of failure. Notice that there exists a difference in the formulation of the exponents of the base e between the equations for repair after the first test and for repair after subsequent tests.

$$PFD(t) = \begin{cases} q+1-e^{-\lambda T_p} + (e^{-\lambda T_p} - q)(q+1-e^{-\lambda(w-T_i)}) \\ \quad \text{for } \{0 < t - T_p \leq TI \ \& \ T_i < w < T_i + T_r\} \\ q+1-e^{-\lambda(TI-T_i-T_r)} + (e^{-\lambda(TI-T_i-T_r)} - q)(q+1-e^{-\lambda(w-T_i)}) \\ \quad \text{for } \{t - T_p > TI \ \& \ T_i < w < T_i + T_r\} \end{cases} \quad (5.4)$$

d) Standby between tests. The quantification is the same as the first standby interval, just different in the model's exponents.

$$PFD(t) = q+1-e^{-\lambda(w-T_i-T_r)} \quad \text{for } \{t - T_p > 0 \ \& \ w > T_i + T_r\} \quad (5.5)$$

It is important to note that the PFD(t) goes through a transient *initial state* before becoming periodically steady (*steady state*). For convenience of formulation, it can be assumed that the initial state comprises the time from $t=0$ up to $t=T_p+TI$. It will be seen later that this elapse of time will be considered as $t=T_{p1}+TI$ for redundant systems.

5.4.2. Inclusion of the effect of automatic diagnostics

First of all, it is important to remember that the total failure rate λ^T in a SIS is split between safe failures λ^S and dangerous failures λ^D (Eq. (1.17)), and that the latter are those only contributing to the PFD. Components for safety systems are usually empowered with automatic diagnostics, which have an important effect on the reduction of their PFD. Their effectiveness is measured with the diagnostic coverage (ε). As a result, the provision of an automatic in-built diagnostics mechanism splits the total failure rate into two failure modes: detected and undetected (Eqs. (1.13-1.14)). For dangerous failures this is:

$$\lambda^D = \varepsilon \cdot \lambda^D + (1 - \varepsilon) \cdot \lambda^D = \lambda^{DD} + \lambda^{DU} \quad (5.6)$$

As previously mentioned, according to Vaurio (1995a), the parameter q in the baseline model (Eqs. (5.2-5.5)) can represent different contributions to the unavailability attributed to detected failures. One of these possibilities is the unavailability attributed to detected failures, assuming that they are detected immediately (i.e. negligible failure residence time). Therefore, it is possible to reformulate the PFD(t) baseline model in terms of detected and undetected failures,

where the former are modelled as a constant, and the contribution by undetected failures shapes the time dependency of the PFD. Therefore, the constant q of the basic model is substituted by the detected failures share of Eq. (5.6) in the subsequent formulations.

5.4.3. The contribution of independent failures

The model developed here intends to include both independent and common cause failures. Since the contribution of CCF to the PFD(t) is the most complex to model, the contribution for independent failures to the PFD(t) is modelled first. The contribution of CCF is modelled in the next section.

We are applying the C_{MooN} modification factor in order to modify the basic β factor according to the system voting logic, as defined by Eq. (4.5). For parallel systems $M=1$. Thus, the modification factor becomes C_{1ooN} . Applying this concept to Eq. (5.6), the detected and undetected failure modes are divided into independent and common cause failures (in the fashion of Eqs. (1.15-1.16)), and thus the total dangerous failure rate is split into four factors:

$$\lambda^D = (1 - \beta^{DD}) \cdot \varepsilon \cdot \lambda^D + \beta^{DD} \cdot \varepsilon \cdot \lambda^D + (1 - \beta^{DU}) \cdot (1 - \varepsilon) \cdot \lambda^D + \beta^{DU} \cdot (1 - \varepsilon) \cdot \lambda^D \quad (5.7)$$

Note that $\beta = \beta_{MooN}$. The four failure modes are renamed to simplify the formulation originating Eq. (5.8), which would correspond to the dangerous failure share in Eqs. (1.17-1.20):

$$\lambda^D = \lambda^{DDN} + \lambda^{DDC} + \lambda^{DUN} + \lambda^{DUC} \quad (5.8)$$

Eq. (5.7) indicates a difference between β factors for detected and undetected failures (as defined by IEC 61508-6). Nevertheless, some methods (e.g. Hauge et al. 2006a) do not distinguish this difference. In this work we do differentiate them, but the choice of the most relevant value for β is left to the reader's discretion.

The model of the PFD(t) due to the contribution of independent failures, $PFD_{ind}(t)$, is derived by substituting Eq. (5.8) into the baseline model given by Eqs. (5.2-5.5). This is made considering the provisions made in Section 5.4.2 regarding the treatment of detected and undetected failures. The contribution of independent failures is given by Eqs. (5.10-5.13).

Notice that this set of equations represents the $PFD_{ind,i}(t)$ for each single component, being the index i the correspondence to the i^{th} component. As a consequence, the time for first test per component T_{pi} and the parameter w_i would change, being denominated as given in Eq. (5.9). Also observe that it is assumed that the TI is the same for all the redundant components.

$$w_i = (t - T_{pi}) \bmod TI \quad (5.9)$$

PFD(t) model for parallel architectures. Independent failures contribution

Standby before first test: (5.10)
 $PFD_{ind_i}(t) = \lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN} t}$ for $t - T_{Pi} < 0$

Testing: (5.11)
 $PFD_{ind_i}(t) = 1$ for $\{t - T_{Pi} \geq 0 \ \& \ 0 \leq w_i < T_t\}$

Repair: (5.12)

$$PFD_{ind_i}(t) = \begin{cases} \lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN} T_{Pi}} + (e^{-\lambda^{DUN} T_{Pi}} - \lambda^{DDN} T_r)(\lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN}(w_i - T_t)}) & \text{for } \{0 \leq t - T_{Pi} < TI \ \& \ T_t \leq w_i < T_t + T_r\} \text{ first repair} \\ \lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN}(TI - T_t - T_r)} + (e^{-\lambda^{DUN}(TI - T_t - T_r)} - \lambda^{DDN} T_r)(\lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN}(w_i - T_t)}) & \text{for } \{t - T_{Pi} \geq TI \ \& \ T_t \leq w_i < T_t + T_r\} \text{ subsequent repairs} \end{cases}$$

Standby between tests: (5.13)
 $PFD_{ind_i}(t) = \lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN}(w_i - T_t - T_r)}$ for $\{t - T_{Pi} \geq 0 \ \& \ w_i \geq T_t + T_r\}$

5.4.4. The contribution of Common Cause Failure

CCF is usually modelled in probabilistic analysis as an additional component in series with the redundant system (see Section 4.1.3). A fault tree depicting this situation for the PFD of a parallel system of three components is shown in Fig. 5.2. It is thus possible to think about the creation of an individual unavailability time dependent model for the CCF component. Figure 5.3 shows the $PFD_{ind}(t)$ for all the components, incorporating a hypothetical plot of the $PFD(t)$ due to the CCF component, denominated $PFD_{CCF}(t)$.

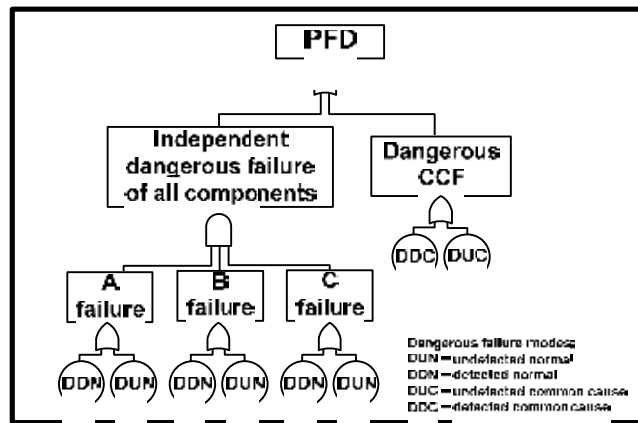


Figure 5.2. Fault tree for PFD of a three-component parallel system

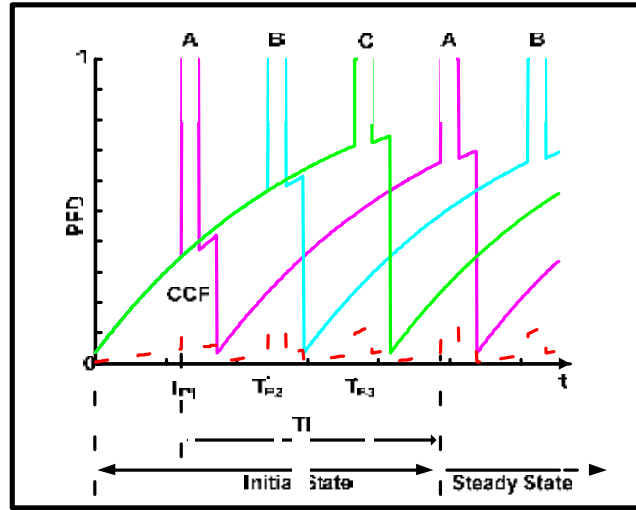


Figure 5.3. System PFD with CCF

Several remarks can be made observing Figure 5.3:

- The CCF unavailability follows the same structure of the mean test cycle (Fig. 5.1): A preliminary interval before first test, followed by test, repair and standby states. Therefore, it can also be defined based on the baseline model.
- The mean test cycle for the CCF is reinitiated every time a component is tested. Thus, the CCF cycle is repeated N times (being N the number of the redundant components) during the TI of one physical component.
- It is clear that after some time the mean test cycle of the CCF will repeat itself periodically, which for formulation convenience has been placed after the time of the first test of the first component plus the test interval ($T_{P1}+TI$), that is when all components have already had their first test and at least one single TI has been completed. After this, all the components contributing to the $PFD(t)$ become periodically stable. The same happens with the $PFD_{CCF}(t)$ component. Based on this assumption, it can be said that the time previous to $T_{P1}+TI$ corresponds to the initial state of the $PFD_{CCF}(t)$, and the time afterwards corresponds to its steady state as shown in Figure 5.3.

Following these remarks, the model has been derived for the $PFD_{CCF}(t)$. In order to keep the model simple, CCF is modelled as a single additional component that fails all the redundant channels at once, and the level of redundancy is reflected using the modification factor of Eq. (4.5) proposed by the PDS Method (Hauge et al., 2006a). After modelling unavailability by CCF for two, three and four components in parallel, it was observable that there exists a pattern, which is repeated for N number of components in parallel ($N>2$). This permits to make

generalizations in the formulation of the exponents (of e) for each state of the time-dependent equations. This was the basis for making the definition of the $PFD_{CCF}(t)$ model for N number of components. The model would be the same as the baseline model (Eqs. (5.2-5.5)), including detected and undetected failures as formulated in Section 5.4.3.

Since the $PFD_{CCF}(t)$ periodicity depends on the moment of the first test of the first component T_{p1} plus the test interval TI , the parameter w for the CCF becomes equivalent to the parameter w for that first component being tested, i.e.:

$$w_{CCF} = w_1 = (t - T_{p1}) \bmod TI \quad (5.14)$$

Based on this concept it is possible to formulate the contribution of CCF to the $PFD(t)$, presented in Eqs. (5.15-5.18). Notice that the formulation for standby before first test, repair and standby between tests (Eqs. (5.15, 5.17, 5.18)) is basically the same as in the baseline model (Eqs. (5.2-5.5)), just with reformulated exponents to accommodate the CCF contribution. Nevertheless, the $PFD_{CCF}(t)$ during test time (Eq. (5.16)) required a different formulation considering several factors. First of all consider that during test of one of the components the $PFD_{CCF}(t)$ requires a formulation that takes into account that this component is unavailable. This means that the CCF is only due to $N-1$ components. Therefore the β factor is modified during the duration of the test. For a system with two components, the CCF becomes zero ($\beta=0$), and for a system with $N>2$, the CCF becomes equivalent to the contribution of $N-1$ components (β_{N-1}). In addition, if the difference in time between tests of the components is less than T_t , this implies that it is a simultaneous TS. Thus, no CCF can occur since all the components are unavailable. Therefore, for cases in which $T_{p_{i+1}} - T_{p_i} < T_t$, the $PFD_{CCF}(t)$ is zero.

The complete $PFD(t)$ model is contained in Eqs. (5.9-5.18). It is important to remark again that in each equation the subscript i stands for the i^{th} component being tested or the last tested. Notice that the model makes the following assumptions:

- The system operates on low demand mode.
- Components failure rates are constant and $\lambda t < 0.1$ (i.e. exponential failure distribution).
- Once a component has failed it remains in that state until it is repaired.
- Failures are detected either by automatic diagnostics or testing.
- The automatic diagnostic period is very small ($\ll TI$).
- Testing and repair are perfect: no faults are overlooked, and the item is returned to an "as new" condition or normal state after repair. The test does not degrade the components
- Components are not available during test.
- Repair rates for detected failure rates (by automatic diagnostics) are also close to zero.

- Test and repair times (for detected failures by testing) are constant values (mean values).
- Test and repair times are very small in comparison to the test interval ($T_t \ll TI$; $T_r \ll TI$).

PFDD(t) model for parallel architectures. CCF contribution
Standby before first test: (5.15)

$$PFDD_{CCF}(t) = \lambda^{DDC} T_r + 1 - e^{-\lambda^{DDC} t} \quad \text{for } t - T_{P1} < 0$$

Testing: (5.16)

IF $T_{P_{i+1}} - T_{P_i} < T_t$ (Simultaneous test)

$$PFDD_{CCF}(t) = 0 \quad \text{for } \left\{ t - T_{P1} \geq 0 \quad \& \quad \left\{ \begin{array}{l} 0 \leq w_1 < T_t \\ T_{P_i} - T_{P1} \leq w_1 < T_{P_i} - T_{P1} + T_t \end{array} \right. \right\}$$

IF $T_{P_{i+1}} - T_{P_i} \geq T_t$ (Non-simultaneous test)

 $PFDD_{CCF}(t) =$

$$\left\{ \begin{array}{l} \lambda_{N-1}^{DDC} T_r + 1 - e^{-\lambda_{N-1}^{DDC} t} \quad \text{for } \{0 \leq t - T_{P1} < TI \quad \& \quad 0 \leq w_1 < T_t\} \\ \lambda_{N-1}^{DDC} T_r + 1 - e^{-\lambda_{N-1}^{DDC} (w_1 - (T_{P_{i+1}} - T_{P_i}) - T_t - T_r)} \quad \text{for } \{0 \leq t - T_{P1} \quad \& \quad T_{P_i} - T_{P1} \leq w_1 < T_{P_i} - T_{P1} + T_t\} \\ \lambda_{N-1}^{DDC} T_r + 1 - e^{-\lambda_{N-1}^{DDC} (w_1 + TI - (T_{PN} - T_{P1}) - T_t - T_r)} \quad \text{for } \{t - T_{P1} \geq TI \quad \& \quad 0 \leq w_1 < T_t\} \end{array} \right.$$

Where:

$$\begin{aligned} \lambda_{N-1}^{DDC} &= \beta_{N-1}^{DD} \varepsilon \lambda^D = \beta^{DD} \cdot C_{100(N-1)} \cdot \varepsilon \lambda^D \\ \lambda_{N-1}^{DDC} &= \beta_{N-1}^{DU} (1 - \varepsilon) \lambda^D = \beta^{DU} \cdot C_{100(N-1)} \cdot (1 - \varepsilon) \lambda^D \end{aligned}$$

Repair: (5.17)

 $PFDD_{CCF}(t) =$

$$\left\{ \begin{array}{l} \lambda^{DDC} T_r + 1 - e^{-\lambda^{DDC} T_{P1}} + (e^{-\lambda^{DDC} T_{P1}} - \lambda^{DDC} T_r) (\lambda^{DDC} T_r + 1 - e^{-\lambda^{DDC} (w_1 - T_t)}) \\ \quad \text{for } \{0 \leq t - T_{P1} < TI \quad \& \quad T_t \leq w_1 < T_t + T_r\} \\ \lambda^{DDC} T_r + 1 - e^{-\lambda^{DDC} (T_{P_i} - T_{P_{i+1}} - T_t - T_r)} + (e^{-\lambda^{DDC} (T_{P_i} - T_{P_{i+1}} - T_t - T_r)} - \lambda^{DDC} T_r) (\lambda^{DDC} T_r + 1 - e^{-\lambda^{DDC} (w_1 - (T_{P_i} - T_{P1}) - T_t)}) \\ \quad \text{for } \{t - T_{P1} \geq 0 \quad \& \quad T_{P_i} - T_{P1} + T_t \leq w_1 < T_{P_i} - T_{P1} + T_t + T_r\} \\ \lambda^{DDC} T_r + 1 - e^{-\lambda^{DDC} (TI - (T_{PN} - T_{P1}) - T_t - T_r)} + (e^{-\lambda^{DDC} (TI - (T_{PN} - T_{P1}) - T_t - T_r)} - \lambda^{DDC} T_r) (\lambda^{DDC} T_r + 1 - e^{-\lambda^{DDC} (w_1 - T_t)}) \\ \quad \text{for } \{t - T_{P1} \geq TI \quad \& \quad T_t \leq w_1 < T_t + T_r\} \end{array} \right.$$

Standby between tests: (5.18)

$$PFDD_{CCF}(t) = \lambda^{DDC} T_r + 1 - e^{-\lambda^{DDC} (w_1 - (T_{P_i} - T_{P1}) - T_t - T_r)}$$

$$\text{for } \left\{ t - T_{P1} \geq 0 \quad \& \quad \left\{ \begin{array}{ll} T_{P_i} - T_{P1} + T_t + T_r \leq w_1 < T_{P_{i+1}} - T_{P1} & \text{if } i < N \\ T_{PN} - T_{P1} + T_t + T_r \leq w_1 < TI & \text{if } i = N \end{array} \right. \right\}$$

5.4.5. Quantification of the average PFD

To quantify PFD using the time dependent model its equations are "embedded" in the tree basic events for independent and common cause failures. Observe this effect in Figure 5.4 and compare with Figure 5.2. It is possible to see that, generalizing, the total PFD(t) for parallel redundant systems with N components can be calculated with the generalization made in the following equation:

$$PFD_{TOT}(t) = \prod_{i=1}^N PFD_{ind_i}(t) + PFD_{CCF}(t) \quad (5.19)$$

The point PFD(t_i) can be evaluated in discrete steps for the length of the mission time. The sum of the point values are then divided over the total number of evaluations n to obtain the PFD_{avg}:

$$PFD_{avg} \approx \frac{\sum_{i=0}^T PFD(t_i)}{n} \quad (5.20)$$

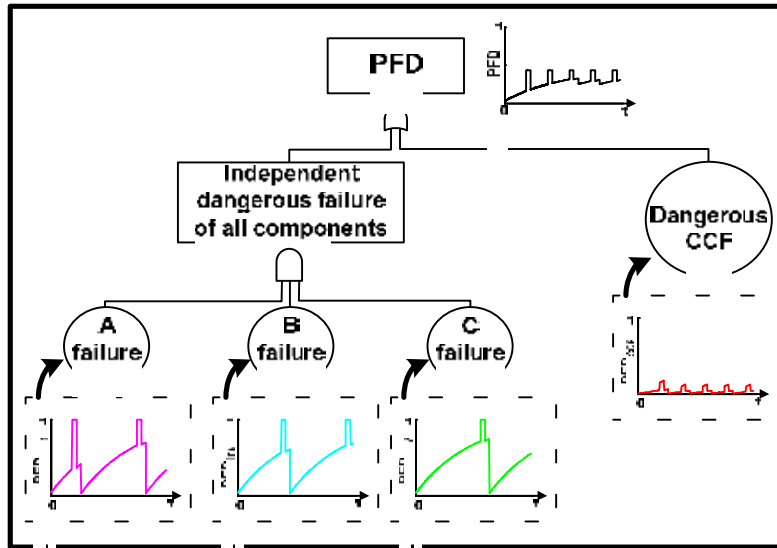


Figure 5.4. Example of use of the PFD(t) model in fault trees

5.5. APPLICATION EXAMPLE

A system corresponding to the fault tree shown in Figure 5.2 is used for application of the PFD(t) model. The data corresponding to the problem is: $\lambda^T=1.9 \times 10^{-6}$; $T_r=8$ hrs; $T_i=1$ hr, $T_I=1$ year. $C_{1002}=1$, $C_{1003}=0.3$. $\beta=10\%$; $\varepsilon=50\%$. The mission time $T=10$ years. It is assumed that $\beta^D=\beta^U$. In order to demonstrate the capability of the model for handling several strategies four different cases have been analyzed, namely:

1. Simultaneous test: $T_{P1}=T_{P2}=T_{P3}=TI/3$
2. Sequential test: $T_{P1}=TI/3$; $T_{P2}=T_{P1}+Tt$; $T_{P3}=T_{P2}+Tt$
3. Non-uniform Staggered test: $T_{P1}=(4/12)TI$; $T_{P2}=(5/12)TI$; $T_{P3}=(6/12)TI$
4. Uniform Staggered test: $T_{P1}=(1/3)TI$; $T_{P2}=(2/3)TI$; $T_{P3}=TI$

The influence of several factors in the system PFD has been analyzed with the data collected: the influence of TS, β factor and diagnostic coverage. For this purpose, and according to Figure 5.4, the total PFD(t) is calculated with Eq. (5.21). The point PFD(t_i) has been evaluated in steps of 1 hour for the length of the mission time, and then the PFD_{avg} was calculated using Eq. (5.20).

$$PFD_{TOT}(t) = [PFD_{ind1}(t) \cdot PFD_{ind2}(t) \cdot PFD_{ind3}(t)] + PFD_{CCF}(t) \quad (5.21)$$

5.5.1. Influence of the test strategy

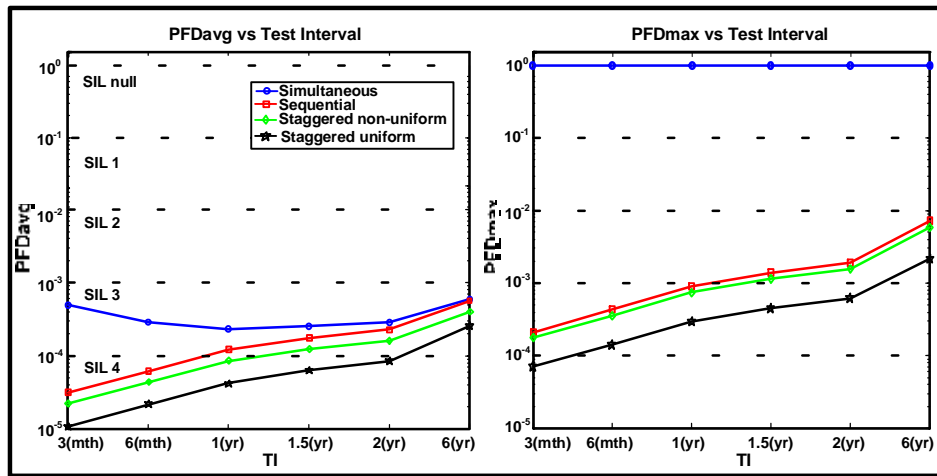


Figure 5.5. PFD for different TI+TS combinations

Plots of the PFD_{avg} and PFD_{max} for different test intervals are presented in Figure 5.5. First of all, observe the behaviour of the simultaneous TS. When increasing TI (lower frequency), the PFD_{avg} improves. This strange phenomenon is the effect of taking out of service the system less frequently. The right-hand illustration remarks that PFD_{max}=1, since all the components have to be taken out of service at the same time. There is, however, a point where further increment of TI affects the PFD_{avg} negatively (as in any other TS it does). Notice in contrast that simply changing the difference of test between components from 0 to barely T_t , i.e. changing the TS from simultaneous to sequential, provides considerable benefits. Firstly and most important, it makes the PFD_{max}<1, which allows the system to be kept in service while testing one component

at a time. Secondly, there is a dramatic improvement in PFD_{avg} (e.g. 79% PFD_{avg} reduction when $\text{TI}=6$ months).

Focus now on the staggered strategy. It is observable that staggering the tests between components reduces further the PFD_{avg} . Firstly, notice how staggering the test decreases the PFD_{avg} in comparison with the sequential strategy (e.g. 29% when $\text{TI}=1$ year). Secondly, observe how spreading uniformly the T_{PI} of the components over the test interval (uniform staggering) improves remarkably the PFD_{avg} (50%). The shift from sequential to non-uniform staggering even increases the SIL achieved by the system (when $\text{TI}=1$ year).

Finally, reading Figure 5.5 from right to left, focusing on the three non-simultaneous strategies, it is clear that a decreasing TI (higher test frequency) improves consistently the PFD_{avg} . Even sometimes better SIL figures can be achieved. It is in general clear that distributing the test events along the TI more informally diminishes the maximum PFD, and in consequence the PFD_{avg} .

5.5.2. Influence of CCF

The improvements in terms of PFD observed above have been achieved simply by modifying the TS, without addressing changes in the system itself. Additionally, this application case addresses the changes in β factor and diagnostic coverage (ϵ). The options being analyzed are referred to by their combination of β factor and diagnostic coverage values as β/ϵ . Five combinations have been tried (for $\text{TI}=1$ year), which results are presented in Figure 5.6. The first three cases in the figure are for a changing β with fixed diagnostic coverage ($\epsilon=50\%$). It can be seen how the increment in the β factor affects directly the PFD_{avg} . Notice that in the ideal case where there is no CCF ($\beta=0\%$), the PFD_{avg} is much lower than for the other two cases. Take as an example the uniform staggered test case. Increasing from $\beta=0\%$ to $\beta=5\%$, raises the PFD_{avg} 438 times (a result not shown, but significant, is that going from $\beta=0$ to $\beta=1\%$ makes the PFD_{avg} larger by 80 times). Changing from $\beta=5\%$ to $\beta=10\%$ represents a higher PFD_{avg} by 100%. Analyzing the figure it could be supposed that for devices with higher total failure rate than those used for the present case, a larger β factor may even imply to lose one SIL level.

The high influence of CCF over the $\text{PFD}(t)$ can be observed closely in Figure 5.7. This figure presents the $\text{PFD}_{\text{ind } i}(t)$ per component plus the $\text{PFD}_{\text{CCF}}(t)$ contribution (black dotted line), and the total and average PFD for one system with no CCF and others with different values of CCF. Note the large increment of $\text{PFD}(t)$ when incrementing the β from 0% to 1%. The significant influence of the CCF contribution is easily noticed. It can be seen that for $\beta>0$ the system $\text{PFD}(t)$ practically has the same shape as the CCF component. These figures suggest that the

CCF becomes the dominant factor in the PFD. It is thus possible to conclude that the PFD_{avg} is quite sensitive to changes in the share of CCF rate.

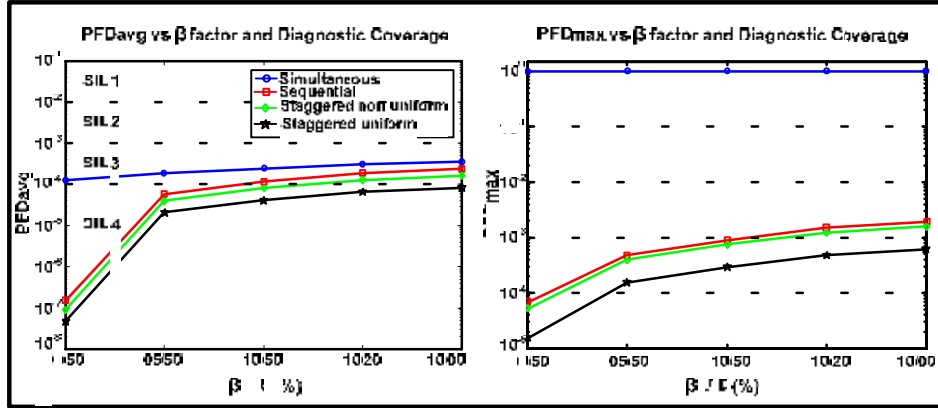


Figure 5.6. PFD for different β/ϵ combinations

The fundamental influence of the CCF contribution can be finally demonstrated with a measure of importance. Consider the Fussell-Vesely measure of minimal cut set importance. This important measure is defined simply as the probability of occurrence of cut set i given that the system has failed:

$$I_i = \frac{P(C_i)}{Q_{SYS}(q(t))} \quad (5.22)$$

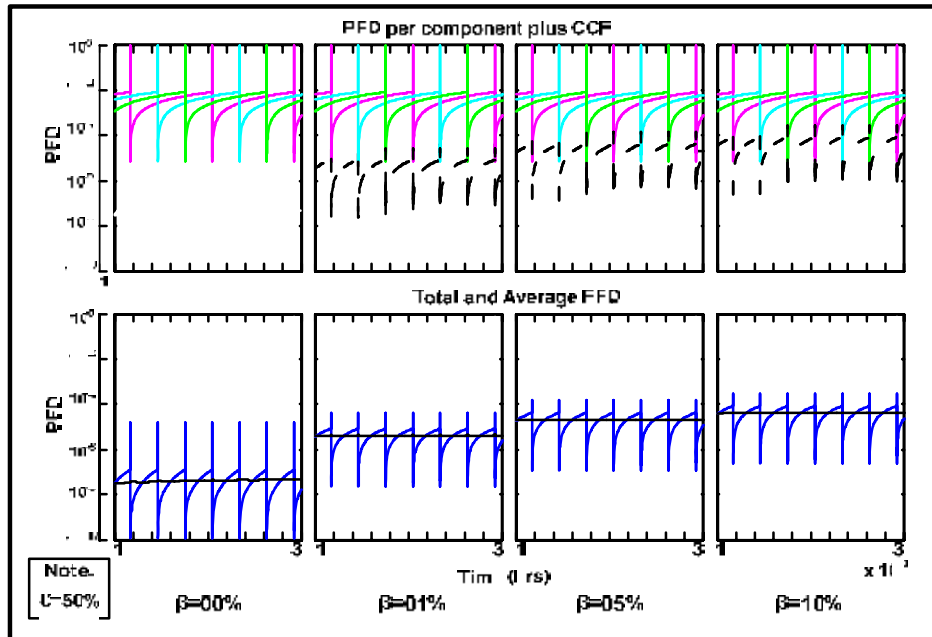


Figure 5.7. PFD for different β factor

If we consider the fault tree of Figure 5.4, where the top event is produced either by the simultaneous independent failure of the three components (where $\varepsilon=0$ for example just to simplify the analysis) or the CCF of all of them, then the importance measure would be:

$$I_{CCF} = \frac{PFD_{CCF}}{(PFD_{ind})^3 + PFD_{CCF}} \quad (5.23)$$

Since $(PFD_{ind})^3 \ll PFD_{CCF}$, then $I_{CCF} \approx 1$; or 100% importance for the CCF cut set. In a numerical example, take the case where the β factor is 1% (remember $\beta^* = \beta \cdot C_{MooN} = 0.003$). For the failure rate of 1.9×10^{-6} : $PFD_{CCF} = 5.7 \times 10^{-9}$ and $(PFD_{IND})^3 = 6.8 \times 10^{-18}$. As it is obvious the importance measure for the CCF basic event is almost 100%, i.e. CCF is the dominant cut set in the fault tree, even for a β factor as low as 1%.

5.5.3. Influence of the diagnostic coverage

The third factor explored in the case study was the diagnostic coverage (ε) of the components. The last three cases in Figure 5.6 correspond to a changing ε with fixed β factor ($\beta=10\%$). For the uniform test staggering case, the improvement obtained in PFD_{avg} from increasing the ε from 0% to 20% and to 50% are reductions of 20% and 38% respectively. It can be seen that the influence of ε is not as prominent as that exerted by the β factor. However, it is still important. Figure 5.6 shows that a better ε may mean the difference between achieving a higher SIL. The relative difference of PFD_{avg} sensitivity between the β factor and the diagnostic coverage is easily observed in Figure 5.8. It is easy to see the changing PFD_{avg} for different values of these measures. While decreasing β in orders of unity makes quite large improvements to the PFD_{avg} , increments in order of tens of ε achieve far more modest gains. It is important, however, not to underestimate the huge positive impact that diagnostic coverage has for system safety.

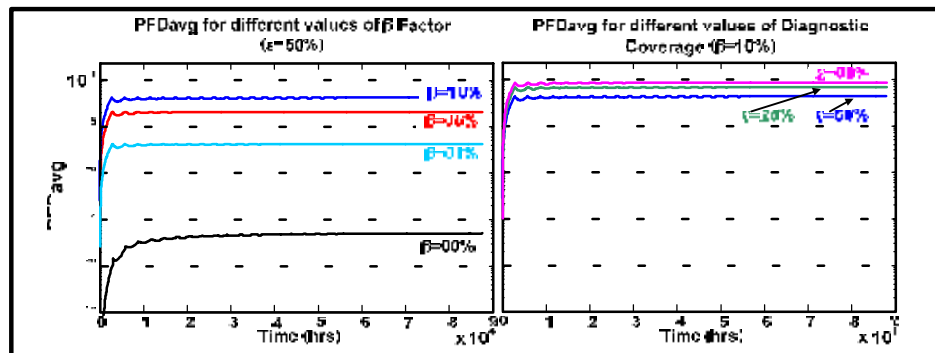


Figure 5.8. $PFD_{avg}(t)$ for different values of β and ε

Further explanation on why the effects of increasing diagnostic coverage are much lower than the effects of decreasing CCF (i.e. the β factor) can be found by observing the fault tree of Figure 5.2. It can be observed that the CCF basic events (detected and undetected) are at the upper level in the tree just next to the top event; i.e. they are first order cut sets. Therefore, any small change in CFF will affect directly the top event. Consider also the Fussell-Vesely measure of minimal cut set importance quantified for the CCF basic event at the end of the previous section. Even for a β factor as low as 1% the CCF has an importance of almost 100%. On the other hand, diagnostic coverage ε reduces the contributions to total PFD when increased. Its effects do reduce the contributions to the probability of failure in all the basic events, but the CCF events are still prominent as first-order cut sets of the fault tree. Its prominence is such that, as it will be seen in next chapter, the total PFD_{avg} of a MooN redundant subsystem (where $M < N$) can be conveniently approximated by simply quantifying the total contribution of CCF to PFD.

5.6. THE MULTI-OBJECTIVE PROOF TESTING POLICY PROBLEM

The second part of this chapter is focussed on the optimization of proof testing policies for Safety Instrumented Systems using the $PFD(t)$ model. As previously mentioned, a proof testing policy includes both the Test Interval and the Test Strategy TI+TS. Any change in the policy impacts on the three system objectives PFD_{avg} , STR and LCC. The test strategies explored for optimization are the sequential and the staggered test. In addition, the difference of staggering can give uniform and non-uniform strategies. Therefore, the degree of distribution of the test events of different components along the test interval of the staggered test strategy provides an additional large number of options for the TS. This distribution, formulated in Eq. (5.24), is commanded by a factor K (Eq. (5.25)) that determines the difference of time between the time of first test of the first component (T_{p1}) and the test of the rest of the components.

Remember in addition that the time elapsed since $t=0$ (system first start-up) up to the first test T_p of each component shapes their contribution to the system PFD_{avg} during the initial time. The larger the T_p the bigger the unavailability contribution (recall that in general $PFD=1-e^{-\lambda t}$). This initial time interval is bounded by the start of the system being put in service until the periodic tests settle down in a steady state. Therefore the TS is composed by both the T_p and the factor K . The T_p per component is then determined as follows:

$$T_{p1} = T_p, \quad T_{p2} = T_p + K, \quad T_{p3} = T_p + 2K, \dots, \quad T_{pN} = T_p + (N-1)K \quad (5.24)$$

Where:

$$K = \text{Round}\left(\frac{TI}{N} \cdot P\right) \quad (5.25)$$

This formulation intends to spread the T_{Ps} of the N components along the TI with the same difference of time amongst them, and that this difference of time is bounded between 0 (in reality T_1) and $TI \cdot (N-1)/N$. Note that a higher K (P closer to 1) means higher staggering, while a lower K (P closer to 0) represents lower staggering. This is better explained by an example. Figure 5.9 shows the distribution of test events for a three-component system (where the components are called A, B and C), and $TI=1$ year. The figure illustrates how for $P=0.25$, the three components are tested during the first two months of the TI (with $K=1$ month difference between consecutive tests). In contrast, when $P=1$, the components are tested with a difference $K=4$ months between them (i.e. they are uniformly staggered, since $K=TI/N$). Thus, a higher P means a more uniform staggering. It is important to observe that the multiplication factor P (Eq. (5.25)), which determines the values of K , is the direct decision variable.

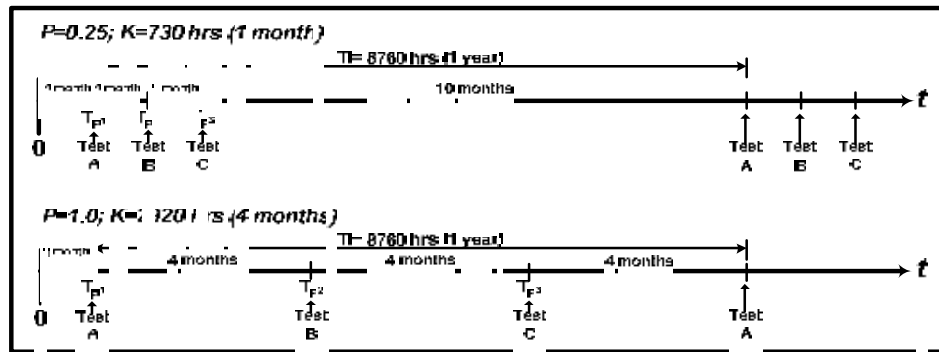


Figure 5.9. Example of T_{Ps} distribution for different values of K

In conclusion, it can be said that the for optimization purposes the proof testing policy depends of three variables: test interval TI , time to first test of first component T_{P1} and multiplication factor P . Given this, it becomes clear that the decision variable vector per subsystem, which intends to solve the decision variables vector in Eq. (2.2), becomes:

$$\mathbf{x} = \{TI, T_{P1}, P\} \quad (5.26)$$

5.7. MODULARIZATION OF FAULT TREES WITH THE PFD(t) MODEL

In the sections where the development of the PFD(t) model is detailed, it has been made clear that all the phases of the mean test cycle have their own contribution to a component's unavailability. In consequence, the model has been empowered with the capacity to handle all

relevant parameters for the quantification of those contributions. Previous analytical models ignore some of those important contributions; namely the test and repair time contributions and the difference in testing staggering. This indeed happens very notably in the method suggested by IEC 61508 Part 6, as Signoret et al. (2007) has also pointed out. For comparison purposes this lesser degree of accuracy in the modelling may be acceptable at a preliminary design stage. However, for quantification and optimization of the impact of different testing policies over the final PFD this approach would be unsuitable. As pointed out by Dudit et al. (2008) “*the saw tooth nature of the instantaneous unavailability cannot be handled properly by using averaged values*” (such as those given by the IEC 61508-6 equations). Hence, the importance of the development of the PFD(t) model presented here.

Section 5.4.4 demonstrated how the PFD(t) model can be integrated in Fault Tree Analysis at redundant subsystem level. For quantification of PFD at system level the same principle applies, but some additional considerations are worth of mention. A large fault tree with several subsystems can be seen as a collection of independent modules, each module representing for example a single subsystem. These modules can be solved separately and their result combined to complete the analysis of the full fault tree (Chatterjee, 1975). This is actually a technique called modularization of fault trees. The modular approach has been applied in both static and dynamic fault trees, as presented by Gulati & Dugan (1997). These authors define the modular approach as one where a fault tree is divided into independent sub-trees (sub-trees that do not share inputs). Each sub-tree can be solved by different solutions techniques. Several methods have been proposed to find the modules (i.e. independent sub-trees) of large trees, such as Kohda et al. (1989) and Dudit & Rauzy (1996). Gulati & Dugan (1997) mention the latter as the most efficient and simple algorithm. They, for example, proposed to convert fault trees module into Binary Decision Diagrams for solution of static fault trees and into Markov chains for dynamic fault trees.

The modular approach makes possible to use different solution methods that make more efficient or easy to solve a fault tree. It also permits empowerment of the fault tree with new capabilities that it usually lacks; i.e. to model time dependencies. For this reason we use here this approach, and propose to incorporate the PFD(t) at subsystem level (i.e. modules), quantifying them in the same fashion as the example of section 5.5. (see Fig. 5.4), for solution of the system-level fault tree. At the same time that the work of this thesis was developed, some other authors have been proposing the use of similar approaches for SIL analysis. Dudit et al. (2006) proposed to incorporate into fault trees multi-phase Markov diagrams for quantification of time dependent unavailability (they prefer to use the term *probability of not functioning on demand* rather than PFD) in several instants of time and average in the same way as proposed in

Eq. (5.20). They illustrated their methodology with an application to a HIPS in Signoret et al. (2007) and further developed their method in Duduit et al. (2008). The method proposed in those articles is quite sophisticated, but too complex for solution of an optimization problem. The problem of exponentially growing complexity in Markov models has been already discussed in Section 1.6.

In this work it is therefore proposed to use the PFD(t) model embedded into the subsystems of the system-level fault trees for solution of complete redundant subsystems, as demonstrated in Figure 5.4 (for one single subsystem). The scope of this thesis does not require the use of any special methodology for identifying modules within the fault trees, since the modules correspond to subsystems easily identifiable. This might be necessary, however, if the size of the modelled SIS increased to several safety functions in future research. Quantification of PFD_{avg} by the concept proposed here is further illustrated in the application case of Section 5.10.

5.8. SPURIOUS TRIP RATE MODEL

It was mentioned in Section 5.1.3 that one of the adverse effects of proof testing includes test-induced spurious activations. This has been also discussed by Lundteigen & Rausand (2008b) in their recent monothematic study of spurious activations in SIS. This is a non-negligible contribution that is usually ignored. Therefore, the STR model presented here includes it. As a consequence, the model for quantification of the STR of a component includes the spurious trips caused by internal failures STR_{λ} (Eqs. (4.10-4.11)) plus the test-induced component STR_{test} :

$$STR_T = STR_{\lambda} + STR_{test} \quad (5.27)$$

On the other hand, STR_{test} is estimated by Eq. (5.28). This was developed based on a general equation formulated by Kim et al. (1994), where some additional guidance can be found to estimate the parameter P_{r-trip} . A similar expression was obtained by Cepen et al. (1994).

$$STR_{test} = \frac{1}{TI} \cdot P_{r-trip} \quad (5.28)$$

Where P_{r-trip} can be estimated by:

$$P_{r-trip} = \frac{N_{trips}}{N_{tests}} \quad (5.29)$$

It is important to take into account that proof testing of every single component of a system can contribute to the test-induced STR. Fig. 5.10 shows how the test-induced STR can be introduced as a basic event into a fault tree of a two-component system.

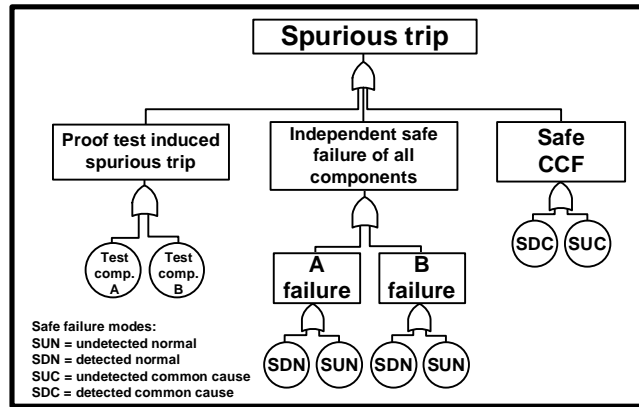


Figure 5.10. Fault tree for STR of a two-component system

5.9. ELITIST NON-DOMINATED SORTING GENETIC ALGORITHM II

One of the most competitive algorithms developed so far is the NSGA-II. Coello-Coello (2006) stated that it is a highly efficient algorithm, with such a good performance that it has become “*a landmark against which other multi-objective evolutionary algorithms have to be compared*”. Greiner et al. (2003) made a comparative study of SPEA2, NSGA-II and NSGA-II with controlled elitism for application to safety system design, and found that the last option gave the best overall average results.

The Non-dominated Sorting Genetic Algorithm NSGA-II was introduced in Deb et al. (2000, 2002). A suggested improvement for diversity preservation called controlled elitism was subsequently issued in Deb & Goel (2001). The NSGA-II is a second-generation algorithm. The characteristic feature of this generation is the implementation of elitism in order to preserve the best individuals for subsequent generations. The NSGA-II has the following main features:

- Ranking based on non-domination sorting.
- Crowding distance metric as explicit diversity-preserving mechanism.
- Implementation of elitism by preserving elitist solutions in the population (using dominance-depth ranking; no usage of external population).
- Diversity preservation and elitism reinforced by a Crowded-Comparison operator.
- Optional controlled elitism for promotion of lateral diversity.

NSGA-II eliminates the necessity of introducing a user defined niche size for sharing, replacing the sharing function by a crowded-comparison operator. It also has the advantage of elitism, and it is supposed to reduce the complexity of the non-dominated sorting. Its elitist mechanism also

eliminates the need for a second archive (of elitist solutions) and its complexities. Its reinsertion strategy (selection for survival) is also simple: a $(\mu+\lambda)$ scheme (Adra, 2007).

5.9.1. Non-domination sorting for ranking

The method determines for each member of the population to which dominance front F_i they belong to (i.e. dominance depth). It then assigns the individual's rank (which is also the fitness) according to the belonging front (non-domination level). In broad terms, the non-dominated set of the population is identified and made front F_1 ($rank=1$). It is then disregarded from the population. A new non-dominated set is identified and assigned front F_2 ($rank=2$). Then this set is also disregarded. The operation continues iteratively until all individuals are assigned to a front F_i and ranked (Deb, 2001). This ranking method is illustrated in Figure 5.11. Compare the difference in ranking with MOGA (Fig. 3.3). The procedure by which the algorithm is implemented is fully described in Appendix B (Section B.5.1).

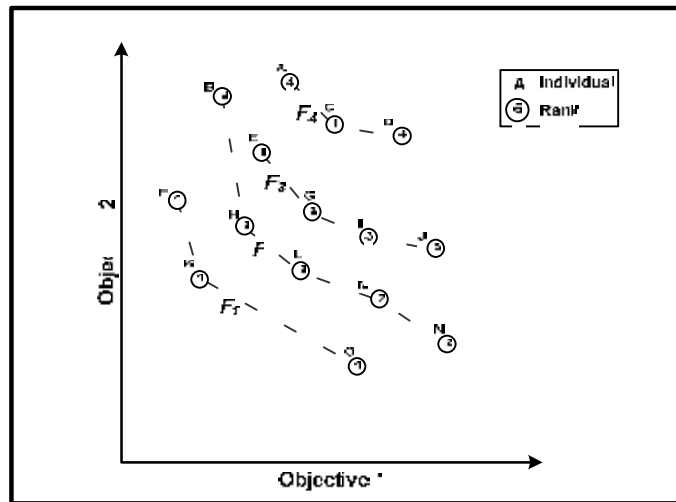


Figure 5.11. Non-dominated sorting ranking in NSGA-II

5.9.2. Crowding distance density estimation

The population density around a particular solution i is estimated by the average distance of the two solutions at either side of i along each of the objectives. This is called the crowding distance d_i , which is an estimate of the parameter of the cuboid formed by the vertices fixed by the nearest neighbours. The measure d_i represents the space not occupied by any other solution around i (Deb, 2001), as it can be seen in Figure 5.12. The crowding distance d_i represents a measure of population density around i . Therefore, the smaller d_i the more crowded its surrounding environment. See section B.5.2 for further details.

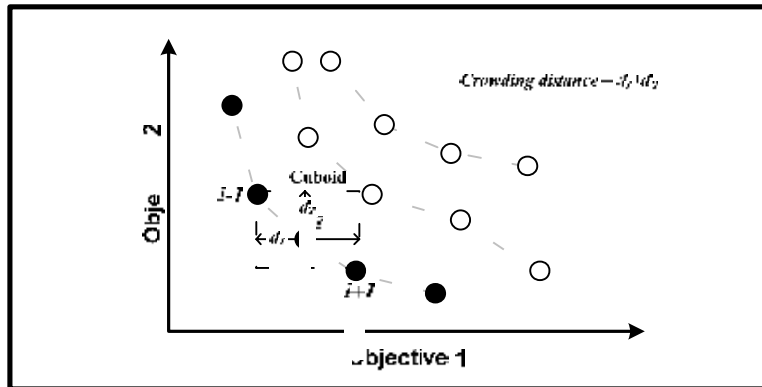


Figure 5.12. Crowding distance calculation

5.9.3. Crowded-comparison operator

Selection for variation is performed using a crowded-comparison operator. The operator performs tournament selection in a similar fashion than the conventional operator described in Section 2.4.3.3. However, besides comparing couples of individuals by ranking (in this algorithm ranking=fitness), it compares their crowded distance for tie-breaking. The algorithm is called Crowded Tournament Selection. It works as follows: When comparing two solutions i and j , solution i is preferred over j if it has a better (lower) rank. If both i and j have the same rank, i is preferred over j if it has a higher crowding distance ($d_i > d_j$). The crowded-comparison operator replaces the sharing function approach used in other GAs, such as MOGA and the first version of NSGA, for diversity promotion. Thus, it does not require any user defined parameter for diversity preservation. The crowding distance is normally calculated in the objective space. Notice that the crowding-comparison operator is applied twice in the algorithm: during the selection for survival (steps 3-4 in next section) and the crowding tournament selection (step 5).

5.9.4. The complete algorithm

The flowchart of the NSGA-II algorithm as proposed in Deb et al. (2002) is presented in Figure 5.13. The algorithm works with a population of fixed size N_{pop} . After creating a random initial parent population \mathbf{P}_0 , each solution is given a fitness value (rank) according to non-dominated sorting. An initial offspring population \mathbf{Q}_0 is created based on traditional binary tournament selection, recombination and mutation. This is then fed to the GA generational loop, which works as follows:

1. Both populations \mathbf{P}_0 and \mathbf{Q}_0 are combined in a population \mathbf{R} (size $2 \cdot N_{pop}$).
2. Ranking is then carried out by sorting the individuals by non-dominated sorting creating several fronts F_i for all i individuals.

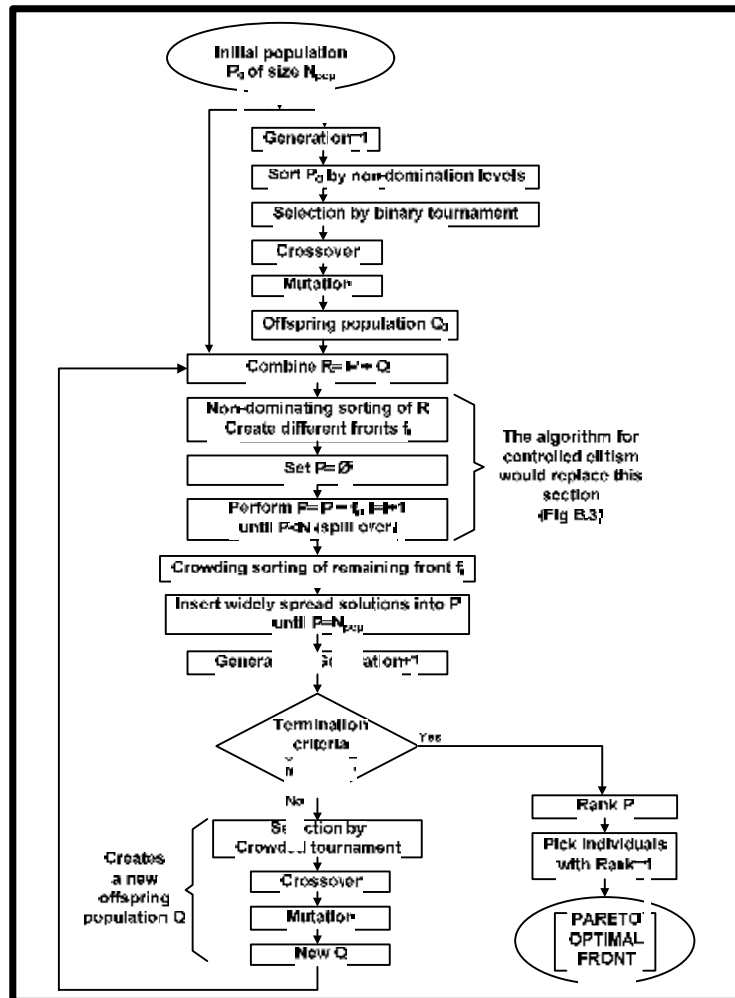


Figure 5.13. NSGA-II algorithm

3. A completely new parent population \mathbf{P}_{t+1} is created and set to $\mathbf{P}_{t+1} = \emptyset$. Then it is populated by adding up the individuals from the fronts, starting progressively from the first front F_1 (better rank=1), the front F_2 and so on until a spillover occurs ($\mathbf{P}_{t+1} > N_{pop}$). This is what enforces the elitist mechanism. The individuals of the front that spilled over are deleted from \mathbf{P}_{t+1} .
4. The entire population of this front (which spilled over) is sorted by crowding distance d_i (the diversity-preserving mechanism), and the individuals with wider spread (larger crowding distance) are inserted into \mathbf{P}_{t+1} until $\mathbf{P}_{t+1} = N_{pop}$. This completes the new parent population \mathbf{P}_{t+1} . Observe that steps 3 and 4 carry out the selection for survival mechanism. This is at the heart of the NSGA-II procedure (illustrated in Figure 5.14). The iterative program could be finished here if the termination criterion were met (as shown in Fig. 5.13).

5. From P_{t+1} a new offspring population Q_{t+1} is created by crowded tournament selection. This executes the selection for variation operation.
6. Crossover and mutation is applied. The new populations P_{t+1} and Q_{t+1} are then used to feed a new generation (to make $R_{t+1}=P_{t+1}+Q_{t+1}$). Goes back to step 1.

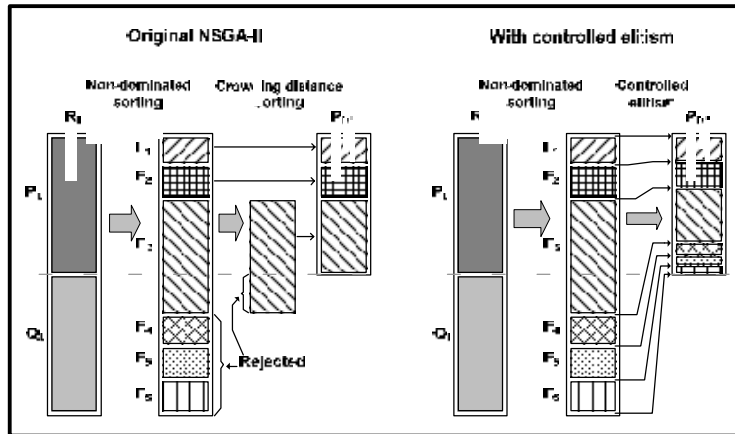


Figure 5.14. Selection for survival; original NSGA-II and with Controlled Elitism (Deb et al., 2002; Deb & Goel, 2001)

5.9.5. Controlled elitism

Emphasis over elite solutions is made twice by NSGA-II: one in the tournament selection and second in selection for survival. This can result in solutions not belonging to non-elite fronts being deleted too soon. This compromises the balance of exploitation versus exploration. The crowding tournament selection will promote diversity along the non-dominated front. However, *lateral diversity* will be lost. This is the diversity lateral to the non-dominated fronts. Therefore, in order to preserve the good balance, it is necessary to ensure good diversity both along the Pareto-optimal front and also lateral to the front (see Fig. 5.15).

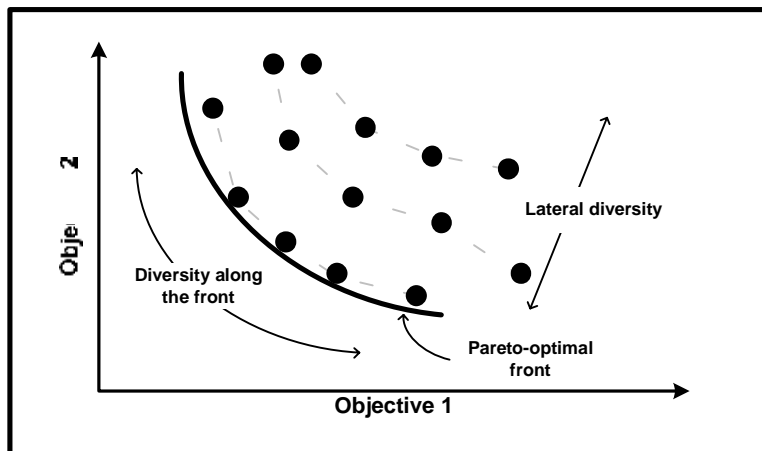


Figure 5.15. The meaning of lateral diversity

Deb & Goel (2001) conceived a mechanism that intends “to maintain a predefined distribution of the number of individuals in each front” (not only the elitist ones), and thus foster lateral diversity (at the time the NSGA-II promotes diversity along the front). This is made by restricting “the number of individuals in the current best non-dominated front adaptively”, allowing individuals of diverse non-dominated fronts to be integrated into the new population. The Controlled Elitism mechanism of selection for survival is compared against the original NSGA-II in Figure 5.14. The maximum number of individuals of the i^{th} front ($i=1,2,\dots, K$) N_i permitted in the new population is restricted by:

$$n_i = N_{pop} \frac{1-r}{1-r^K} r^{i-1} \quad (5.30)$$

Where N_{pop} is the size of the population and r is the reduction rate ($r < 1$). With this, “each front is allowed an exponentially reducing number of solutions”. Notice that the allowable number of individuals from the first front is the highest, and it decreases progressively with each front. The full procedure to implement the algorithm is detailed in Section B.5.3. Figure 5.13 indicates which part of the original NSGA-II would be replaced with the controlled elitism mechanism.

As Deb & Goel (2001) discuss, the reduction rate r is important in the balance exploration versus exploitation, since it determines the extent of exploration allowed: the smaller r the larger exploration and vice versa. However, the parameter r is also problem-dependent, becoming an additional parameter to tune for the GA. In the test problem given in Deb (2001), a $r=0.65$ keeps a good exploration-exploitation trade-off. This may be a good value for start tuning it.

5.10. APPLICATION CASE

5.10.1. Description of the problem and approach

In order to illustrate the overall approach, the proof test optimization is applied to a protection system against high pressure and temperature of a chemical reactor (Fig. 5.16). The system is composed of four subsystems: Temperature transmitter (TT), pressure transmitter (PT), logic solver (LS) and final control element (FC). Upon detection of either high temperature or pressure the safety system cuts the reactor supply off in order to prevent a runaway reaction. Each subsystem is parallel redundant. All data are presented in Table 5.1.

The optimization cases presented here consider the possibility of using sequential or staggered test. As concluded above, partial testing provides the opportunity for optimization. International standards do not present constraints for partial testing, in addition they do not provide sound

guidance about the setting up of TIs. Therefore, the proof test optimization problem is addressed by optimizing the test policy for each single subsystem. This means that 12 variables (3 per subsystem: TI, T_{P1} and P) compose the decision variables vector in Eq. (5.26).

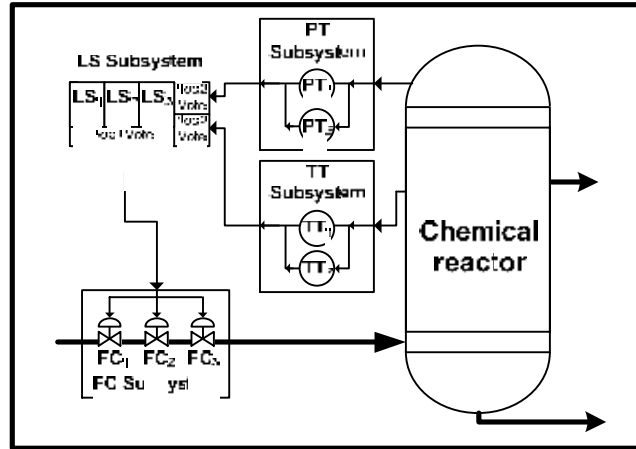


Figure 5.16. Chemical reactor protection system

Table 5.1. Dependability and Lifecycle Cost data

Instrument	Safety PLC	Standard PLC	Electronic PT	Press. Switch	Smart TT	Temp. switch	Air Opt. valve
λ_{Safe} ($\times 10^{-6}/h$)	3.46	3.94	2.16	6.81	5.05	9.22	3.94
λ_{Danger} ($\times 10^{-6}/h$)	0.0036	4.81	1.90	4.11	0.348	7.60	3.35
λ_{Total} ($\times 10^{-6}/h$)	3.492	8.75	4.06	10.92	5.398	16.82	7.29
ϵ_{Safe} (%)	100	45	56	10	100	10	00
ϵ_{Danger} (%)	81.25	60	51.1	10	7.5	10	25
$C_{purchase}$ (\$)	3000	2000	2300	500	2560	500	6940

Lifecycle cost data:

Design/install/commissioning PLC= 10,320 (\$)

Maintenance PLC= 960 (\$/event)

Test PLC= 240 (\$/event)

Design overall instrumentation= 3,060 (\$)

Installation/commissioning per instrument=600 (\$)

Maintenance per instrument= 240 (\$/event)

Test per instrument= 60 (\$/event)

Repair cost per instrument & PLC= 60 (\$/h)

Spares per repair= 25% component cost per event

Cost safety PLC rack: 31000 (\$)

Cost standard PLC rack: 20500 (\$)

Loss of production= 2,000 (\$/h)

Start up cost= 1800 (\$)

Catastrophic loss= 150×10^6 (\$)

SIS operational life= 15 (years)

Discount rate=0.05

Other data:

$\beta=10\%$ (for all instruments, both failure modes)

$\beta=5\%$ (smart TT, safe failure only)

$T_r=8$ (h)

$T_i=1$ (h)

$C_{1002}=1.0$

$C_{1003}=0.3$

Shut down time= 24 (h)

Plant risk without SIS= $8.55 \times 10^{-3}/yr$

Three different cases have been analyzed:

- Case 1. SIS with safety instruments: Safety PLC, conventional electronic pressure transmitter, smart temperature transmitter and air operated valve. We use a $P_{r-trip}=1 \times 10^{-3}$. Kim et al. (1992, 1994) suggest a method for estimation of P_{r-trip} , giving examples for some instruments. The value used here has been estimated as a representative average following

values provided by those references.

- Case 2. The SIS uses the same kind of instruments, but with a $P_{r-trip}=0$ (ideal proof test case), in order to draw comparisons with Case 1.
- Case 3. The SIS is composed by standard instruments: Standard PLC, electromechanical switches for pressure and temperature and air operated valve. $P_{r-trip}=1 \times 10^{-3}$.

5.10.2. Problem modelling

Modelling of the dependability objectives has been discussed in previous sections of this chapter, so only some minor problem-specific details are added here. System dependability is quantified by Fault Tree Analysis. The fault tree for STR follows the philosophy shown in Fig. 5.10. The fault tree for PFD is presented in Figure 5.17. Observe that the subsystems' sub-trees are not fully developed as in Figure 5.2. In Figure 5.17 the basic events are the failures at subsystem level, given that they are considered as independent modules and quantified following the philosophy illustrated in Figure 5.4. Eq. (5.31) gives the expression that calculates the system PFD(t). Each term in the equation indicates correspondence to a subsystem by a subscript. The instant values are averaged as indicated in Eq. (5.20) to get the PFD_{avg} .

$$PFD(t) = PFD_{LS}(t) + (PFD_{PT}(t) \cdot PFD_{TT}(t)) + PFD_{FC}(t) \quad (5.31)$$

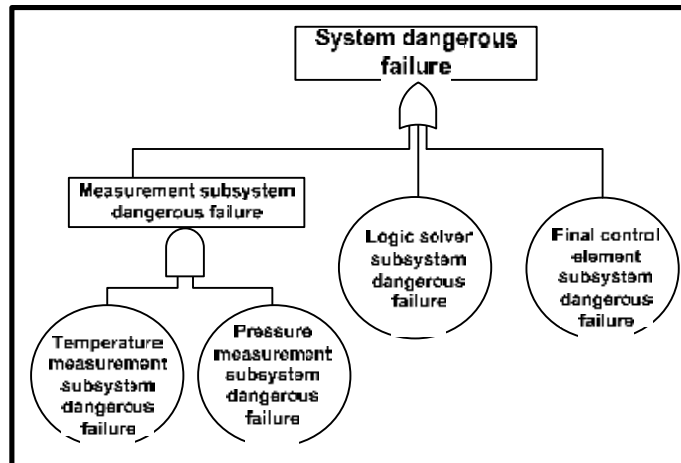


Figure 5.17. Fault tree for quantification of PFD

The LCC of the system is calculated based on equations given in Chapter 4 (Eqs. (4.12-4.22)). From Eq. (4.12), the operational costs relevant to proof testing include test and corrective maintenance during the system lifetime:

$$LCC_{T\&CM} = (C_{T\&CM} + C_{RISK}) \cdot PVF \quad (5.32)$$

Where:

$$C_{T\&CM} = C_T + C_{CM} \quad (5.33)$$

C_T and C_{CM} are estimated using Eqs. (4.16) and (4.17). In this work we are provided with the cost of test per event (alternatively it could also be determined by Eq. (5.34)). The repair cost per hour is estimated with Eq. (5.35).

$$C_i^T = T_i \cdot C_{ht} \quad (5.34)$$

$$C_i^{CM} = (T_r \cdot C_{hr}) + C_{SP} \quad (5.35)$$

Notice that the corrective maintenance costs include repair of safe failures as well (Eq. (4.17)). Although proof testing is focussed on dangerous failures detection, it is clear that if safe failures occur they must be repaired as well.

On the other hand, it is important to consider the impact that proof testing has upon the plant risk cost, which is calculated by Eqs. (4.19-4.22). Proof testing has both positive and negative effects upon LCC, since the practice of periodic proof testing reduces the dangerous failure risk when improving PFD_{avg} , but it can increment the safe failure risk (through its adverse impact on the STR).

5.10.3. Implementation of the Genetic Algorithm

The optimization is based on the NSGA-II (Fig. 5.13). All the 12 decision variables of the problem are integer. Explicit constraints are included in the decision vector: The TI is bounded between 30 and 365 days (730 for the LS), and T_{ps} between 0 and TI. The multiplication factor P is a number between 0 and 1000 (then multiplied by 1×10^{-3} so that $P < 1$). The total number of potential solutions is 9.27×10^{29} . Given that the decision variables are integer, an integer code is used rather than the classic binary code. This opened the possibility of using genetic operators for both binary and real numbers (with minor modifications), especially for the crossover and mutation operators.

A baseline algorithm setup was used as the initial configuration. Then the algorithm was tuned trying several different schemes. A qualitative comparison was made between pairs of different algorithm setups, based on a good balance of the following factors: convergence (proximity to the real (unknown) Pareto Front), and the diversity along the front (good uniform spread of solutions) and number of individuals in the optimal set.

Only a few of the GA multi-objective optimization studies report use of integer codification. They implement combinations of operators for either binary or real numbers. Coit & Smith (1994, 1996a) used an algorithm similar to uniform crossover and mutation by integer flipping. Billings & Zheng (1995) reported what seems to be two variations of the uniform crossover algorithm, called fixed length and variable length crossover. Mutation was made through a mechanism similar to integer flipping. Weile & Michielsen (1996) proposed special formulas for crossover (similar to blending methods), and utilized addition or subtractions formulas for mutation. Martorell et al. (2000) used single point crossover and flip mutation. Tao et al. (2003) implemented double-point crossover, and performed mutation through addition formulas. Damousis et al. (2004) used multiple point crossover and employed a non-uniform mutation operator. Li et al. (2005) applied formulas similar to blending methods for crossover, and mutated the individuals through integer flipping. Salazar et al. (2006) used a hybrid integer-real code. He used a single point crossover, just permuting the values where the crossover point laid between integer variables. They tried both flipping mutation and a “triangular mutation operator” emulating Gaussian mutation. Some other optimization studies were found, but they do not specify which operators they implemented.

The settings that were tried (in different combinations) are detailed below. The crossover and mutation operators for real numbers that were tested are those formulated by Mulenbein & Schlierkamp-Voosen (1993), described in Section 2.4.3.

- Population size and number of generations: 50/100, 100/50, 200/20, 100/25, 100/40.
- Mutation: Flip mutation at 0.1, 0.2; mutation for real numbers at 0.1, 0.2.
- Recombination: Single point crossover, double point crossover, discrete recombination (uniform crossover), blending crossover.
- Selection: Basic NSGA-II against controlled elitism at $r=0.50, 0.80, 0.65, 0.40$.

In general, it can be reported that a population size larger than 100 provided no significant improvement. Fine tuning could be made through changing the number of generations. Changing the mutation operator (using both algorithms for real and binary codes) did not make any important difference. From all the recombination operators compared, discrete recombination showed a slightly better performance, while a more notable improvement was found with the blending crossover: Better distribution along the optimal front and enhanced exploration since new solutions at the two extremes of the front were found. The most significant enhancement overall was achieved by implementing controlled elitism. It was noticed that this improves the exploration made by the algorithm. Nevertheless, the exploitation can be compromised, and it is only improved if the right reduction rate parameter r is found. This fact, however, increments the complexity of the NSGA-II implementation, since the

parameter r has to be tuned. The improvements achieved in terms of exploration meant that both ends of the optimal front were further pushed, finding new solutions. The distribution along the front was slightly enhanced as well.

The best final combination of parameters was:

- Population size: 100 individuals.
- Number of generations: 50.
- Recombination by blending crossover at 0.7.
- Flip mutation at 0.1.
- Selection using controlled elitism with $r=0.40$.

5.11 DISCUSSION OF RESULTS

Case 1 is taken as the baseline case. The results of this case are presented in Fig. 5.18. In order to demonstrate the benefits of the optimization by GA, the Pareto-optimal front (i.e. optimal set) is compared against the initial population. A specific single solution may have been proposed as an initial candidate solution and introduced into the initial population for comparisons purposes. However, it was considered that the random creation of the initial population (initial set) would contain both good and bad solutions. So this is used as basis for comparison. It is easily observed that the optimal set is superior to the initial set in every objective. The optimal set provides a smooth well distributed set of solutions for the decision maker to choose. It is clear that every single solution provided in the initial set is dominated by the optimal set. Hence, the advantage of implementing the GA-based optimizer is evident.

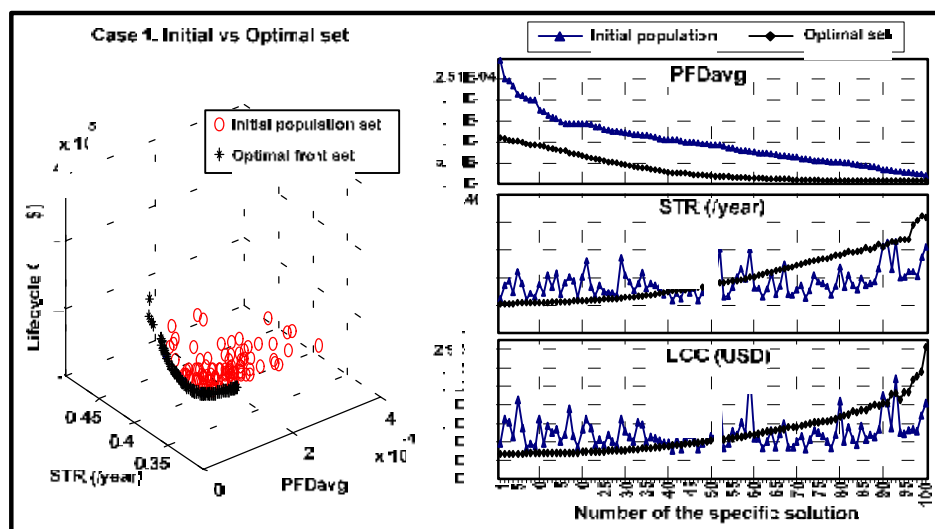


Figure 5.18. Case 1. Comparison of initial and optimal set populations

The plots in Fig. 5.18 (right) compare the performance of both populations by pairs of objectives. Notice that for any PFD_{avg} value, the cost (LCC and STR) in the optimal set is always lower than in the initial set. It shows the benefits of risk reduction comparing again both initial and optimal sets. It can be seen how around solution 50, any reduction in PFD_{avg} has a lower LLC and STR costs (in comparison with the initial set). After that point, further reduction of the PFD_{avg} is more expensive. Notice however, that in every case the PFD_{avg} is lower for the optimal set than for the initial set. Observe though that when the graph is following the increasing solution numbers, gains in terms of PFD_{avg} become more expensive in terms of both LCC and STR. The optimal set shows that there is an area where risk reduction (lower PFD_{avg}) is practically free (low cost increment). Around solution 40, risk reduction becomes much more expensive.

Fig. 5.19 shows the Pareto-optimal fronts for the three cases. The complete results obtained from all three cases are presented in Fig. 5.20. This figures plot the multiplication factor P rather than K , since this is the direct decision variable (see Eq. 5.25). It is very important to notice that a different scale is used for Case 3 in comparison to Cases 1 and 2. Observe the two scales at the top of Fig. 5.20, at the left and right extremes.

Fig. 5.19 corroborates that the PFD_{avg} is consistently in conflict with the STR and LCC. In addition, notice that in Case 1 there is a saturation point in the Pareto-optimal front, where a marginal reduction of PFD_{avg} causes a large increment of both STR and LCC. This starts around solution 80 (also seen in Fig. 5.20). Trying to achieve any further small reduction of PFD_{avg} after this point is very expensive, and it would be justified only to meet some specific safety requirements. Again, notice that from solution 1 to 41 risk reduction cost is more profitable (the cost of improving the PFD_{avg} is less significant).

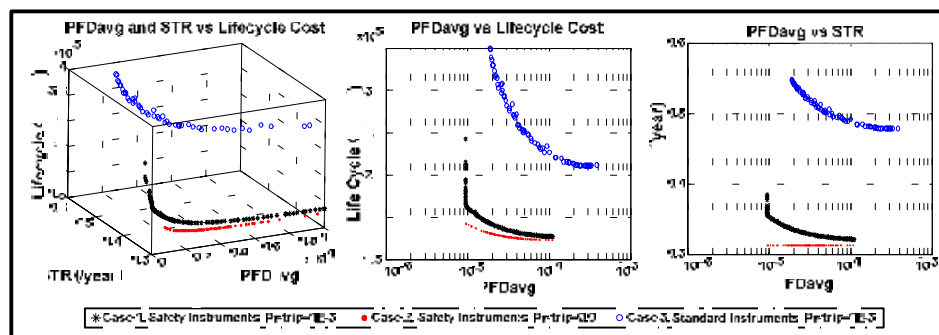


Figure 5.19. Comparison of the optimal front for the three cases

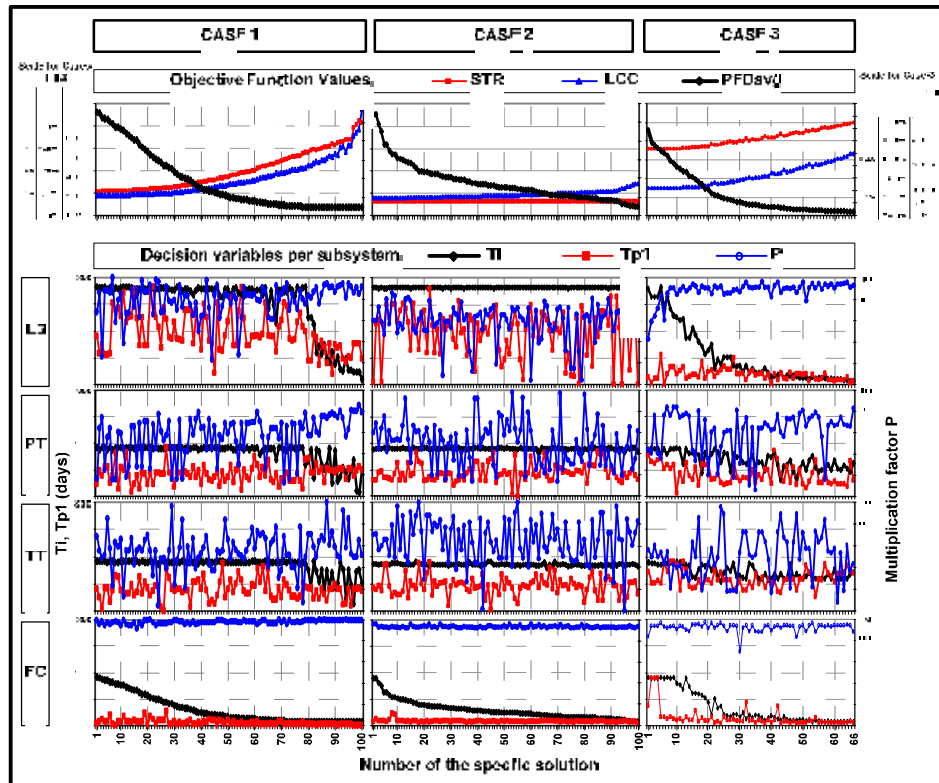


Figure 5.20. Results obtained from all cases

Analyzing the decision variables obtained for the optimal set of Case 1 (Fig. 5.20), the following comments can be drawn. First of all, the test interval TI of every subsystem depends on the reliability specifications of its components. The lower the dangerous failure rate (and higher diagnostic coverage), the higher TI (lower test frequency). Observe that, in general, the TI for the LS subsystem is above 700 days, while for the other subsystems it is below 400 days. On the other hand, for the LS, PT and TT subsystems the T_{p1} and P variables seem not to have a consistent trend. However, the case of the shutdown valves (FC) is notable. The T_{p1} is generally low, meaning they are tested early from the start of the system's operational life. The FC subsystem TI declines steadily along the system PFD_{avg} reduction; i.e. the overall system PFD_{avg} is following the FC's TI trend. This is due to the fact that the FC subsystem is the system's weakest link, given that the valves have the highest failure rate and lowest diagnostic coverage. If we follow the parallel decline of system PFD_{avg} together with the FC's TI , we will notice that once the TI is very low, and cannot be further lowered, the TI of the other subsystems starts decreasing, i.e. lower TI (around solution 80) specially in the LS. This indicates that the improvements in PFD_{avg} are achieved testing the FC components more frequently. Once this frequency cannot be increased, the benefits have to be extracted increasing the test frequency of

another subsystem. This is by natural choice the LS, because the TP and TT subsystems act as redundant subsystems of one another. Also observe that in both the FC and LS subsystems, once the TI has been considerably lowered, the T_{P1} (time to first test) is also decreased.

Note that the behaviour of the multiplying factor P is similar to the one of TI. This means that in order to obtain further reductions in PFD_{avg} the distribution of the test of the components is made more uniformly; i.e. full staggering. Notice that no single value of P goes close to zero, which clearly indicates that sequential testing is completely discarded.

Observe now the performance of the optimization for Case 2 in Figure 5.20. Remember that Case 2 corresponds to the ideal case where the proof testing activity has not adverse effects over the system STR ($P_{r-trip}=0$). In the upper plot it is noticed that PFD_{avg} is still in conflict with the LCC: The safer the system the more expensive it is (although the increment in LCC is not as pronounced as in Case 1). However, it is not in conflict with the STR anymore. Observe that the STR plot is flat. Different from Case 1, there is no saturation point; the test frequency is constantly low for the three more dependable subsystem: LS, PT and TT. Nevertheless, the PFD_{avg} behaviour continues to follow the FC's TI trend: The shutdown valves remain the system weakest link. Referring to Figure 5.19, it is also evident that the LCC is lower. This situation is possible basically because there is no STR increment caused by testing. Thus, the most important conclusion from comparing Case 1 against Case 2 can be drawn: Apart from the natural costs caused by the testing activity itself, it is illusory not to consider the adverse effects of testing, especially for STR in SIS. Testing does have an additional, considerable cost through STR. Thus, it is fundamentally important to consider its adverse effects for decision making.

Finally, consider the results given by Case 3 in Figure 5.20. This case implements the safety function using less reliable components than in Case 1. Notice the difference between Case 3 and Case 1 in the optimal solutions obtained (do not forget to consider their different scales again). The most notable is the case of the LS subsystem. All in all, now the system PFD_{avg} follows both the LS and FC test interval (in Case 1 it did only the FC's TI). This is because now the PLC failure rates (and diagnostic coverage) are not outstandingly lower than the rest of the instruments (as it was in Case 1). Notice as well that the subsystems PT and TT are tested more frequently than in Case 1, despite them backing up each other. Fig. 5.19 clearly shows that the optimal set of Case 3 is always dominated by the one of Case 1. At every value, the cost for a similar PFD_{avg} is higher for Case 3, both in terms of STR and LCC. The acquisition cost for Case 3 system may be lower, but in the long term, for the entire system operating life, Case 1 is much less costly.

Considering test staggering it is important to make some conclusions: Section 5.5.1 showed that there is a definite benefit in PFD_{avg} , since increasing the level of staggering (i.e. uniform distribution of the test events along the TI) reduces PFD_{avg} . First of all, the maximum PFD is reduced with the more uniform distribution of the test events. In addition, the CCF component is tested more frequently; hence its contribution to PFD is actually reduced. It can also be seen in Figure 5.20 that in increment of the variable P (in the LS and PT subsystems in Case 1 and LS subsystem in Case 3) coincides with the increment of STR and also LCC. At the same time, however, the TI is being reduced. Therefore, it is clear that the increment in STR is due to the increasing testing frequency. There is no evidence to suggest that increasing test staggering can affect negatively the STR.

5.12. CONCLUDING REMARKS

This chapter approached the modelling and optimization of proof testing policies for SIS. A model with a new approximation for the time dependent PFD of parallel architectures in SIS was presented here. The model addresses the level of modelling detail required by IEC 61508, since it is capable of evaluating explicitly the effects of the component's diagnostic coverage and the system Common Cause Failure. It is powerful enough to accommodate different testing strategies, i.e. simultaneous, sequential and staggered (with varying degrees of staggering).

An application to a parallel three-component arrangement showed clearly the benefits of staggering the test events of the redundant components of the system, improving the PFD as the time for the test event is distributed more uniformly during the Test Interval. In addition, increments of testing frequency (lower test intervals) affected also positively the PFD. Safety Instrumented Systems usually employ components with very low failure rates. It has been demonstrated that in these cases the CCF has a significant impact on the system PFD. Even a low β factor makes the contribution of the CCF to shape the $PFD(t)$. Practically the value of the $PFD(t)$ becomes the equivalent to the value of the contribution of the CCF. On the other hand, albeit the diagnostic coverage has not as prominent influence as the CCF over the PFD, it is certainly improved when the coverage is enhanced. It is known that the inclusion of a built-in diagnostic mechanism is one fundamental strategy for the improvement of performance of safety systems. To use components with better diagnostic coverage can reduce the PFD in such a way that it can even achieve a higher Safety Integrity Level.

The model was subsequently integrated into a methodology for optimization of proof testing policies with the NSGA-II genetic algorithm for a four-subsystem protective system. The requirements of IEC 61508 and current real testing practice for SIS were taken into account.

Proof testing system policies are evaluated as the combination of test interval and strategy TI+TS. The test strategy includes the time for first test of the components the strategy itself (sequential or staggered), and the level of staggering.

The results obtained show that proof testing is very relevant for achieving and maintaining high levels of system safety integrity. At the system level increments of testing frequency (lower TIs) also affect positively the PFD_{avg} . This is, however, in conflict with the system LCC. Apart from the obvious benefits of the testing activity, it also certainly conveys adverse effects. One of the most relevant is an increment of the STR. It has been demonstrated that ignoring this factor in the dependability model results in overoptimistic assumptions. It is therefore important to consider and quantify them. On the other hand, the results show that the combination TI+TS is very influential on the system safety integrity; especially the TIs of the components with the lowest dependability in the system (high dangerous failure rates and low diagnostic coverage) have even shaped the system PFD_{avg} . The second observed factor is the trend of the test to be fully staggered for lower PFD_{avg} levels. The application case showed the benefits of staggering the test events of the redundant components of the system, improving the PFD_{avg} as the time for the test events are distributed more uniformly along the TI.

The choice of instruments with lower cost may lower the initial acquisition cost of the system, but at the same time they can have the effect of increasing considerably the overall system LCC. This knowledge can be used for decision making regarding system design at early stages of the lifecycle. Nevertheless, testing should not be overexploited. As with design optimization, there is one point where further intensification of the testing activity certainly enhances safety lowering the PFD_{avg} but a very high cost: Very small marginal decrements of PFD_{avg} may have disproportionate costs in terms of both STR and LCC. Considering all of these factors, it is easy to conclude that the optimization methodology implemented offers many benefits to the decision maker, in terms of both the optimization itself and a better knowledge of the safety system behaviour, making it a powerful tool analysis for practical industrial application.

CHAPTER 6

Modelling and optimization of SIS including MooN voting architectures

The developments presented in previous chapters have addressed only redundant systems implemented with parallel redundancies. This has followed the rationale that it was simpler to develop models and optimization techniques for parallel systems and then to extend them to MooN voting systems. This chapter, therefore, approaches the optimization of SIS where inclusion of MooN voting architectures is an available option for subsystem redundancy. It includes optimization of system design and optimization of test policies. The main aim is to explore whether introducing voting architectures as an alternative to simple parallel redundancies allow achievement of a better balance between the system's dependability attributes (PFD_{avg} and STR), and therefore better LCC. Regarding optimization of the testing policies, this chapter also analyzes if introduction of MooN subsystems in the system facilitates the implementation of better testing policies that counteract the negative impact of testing while still giving benefits in terms of safety and cost.

The structure of the chapter is as follows: A preliminary review of concepts and modelling techniques related to MooN systems is provided together with a review of their application in optimization cases. In order to extend the application of the PFD(t) model presented in the previous chapter to MooN architectures, an analysis of modelling of MooN systems with fault trees and their probabilistic behaviour is made, obtaining expressions for quantification of PFD at subsystem level. This is subsequently integrated with the PFD(t) model. A similar analysis is made for extending the STR model used in previous chapters to MooN architectures. The models are then integrated with optimization of SIS in two separate applications: design and testing policies.

6.1. OVERVIEW OF MooN VOTING ARCHITECTURES

A MooN system was defined in Section 1.3 as a system with N units (i.e. components, channels, etc.) in which M out of them are sufficient to initiate the safety function. It requires a minimum of M units to "vote" for the execution of the safety function. It also implies that these M units must be functional for the system to be successful, and that a voting mechanism is provided.

This is equivalent to a k -out-of- n :G with an added voter. There are, however, several diverse terms to designate the same or similar configurations, which can be confusing. A detailed review of terms is presented in Appendix A, Section A.3.1.

6.1.1. Effects of introducing voting architectures

Safety systems must present fault tolerance, mostly against dangerous failures. As it was seen in Section 1.2 this is implemented usually by use of redundancy, where in order to ensure continuity in the execution of the safety function, one failed component's function is taken over by a redundant element. The most basic and used form of redundancy is the parallel structure. However, every component that is added to the system introduces a new chance of failure. This results in new chances of spurious trips. It has been seen in previous chapters that spurious trips can bear a considerable cost. This is a manifestation of the conflict between PFD and STR. For this purpose, voting systems are introduced in the search for the achievement of a better compromise between fault tolerance and spurious trip rate.

In comparison with active standby redundancies, the fact that the redundancy is passive in MooN systems eliminates some inconveniences; i.e. the switchover time, the probability of switch failures and the probability of unrevealed standby-mode failures. In this way a voting system ensures the non-interrupted operation, which is fundamental in safety-critical systems. At the same time, although a non-perfect voter may be a new issue introduced into the system architecture, a voter is usually less complex than a switch mechanism for cold or warm standby systems (and even for some complex couplers of parallel systems). A voter is a fault masking mechanism, simpler than more sophisticated fault tolerance mechanisms that involve fault detection and system reconfiguration, and which have their own integrity issues.

A redundant voting system usually has a higher reliability than a single component (in the high-reliability region, which is customary for safety components), although less than a parallel system. At the same time, increasing the number of voting components increases the MTTF, and raises system reliability (in comparison with other voting systems). A low STR must be a fundamental attribute of a safety system, since it has an impact on the system LCC, and what is very important, the confidence the operator places upon the system. The operator may even be tempted to bypass (or even switch off!) a noisy safety system, negating completely the benefits it should bring to plant safety.

It can be seen that a voting system permits implementation of fault tolerance with a lower STR than a parallel system with the same number of components N . Although it improves the PFD_{avg} compared with a single-component system, compared with a similar parallel system it, however,

does not. Nevertheless, it contributes to ensuring continuous operation of the safety system.

Considering what has been described, the option of choosing amongst voting systems could permit an additional reduction of STR while implementing fault tolerance. This may represent an increment in the acquisition cost, but a decrement of STR costs. It is thus a question of cost balance to achieve a positive impact on the system LCC. Due to the increment on PFD_{avg} (in comparison to a similar parallel system), a voting architecture could have a negative impact on the test intervals. Also remember, as we found in the previous chapter, that proof testing may also have a negative impact on STR. It is therefore a matter of research to determine whether the benefits of introducing voting architectures regarding STR outweigh the disadvantages of higher PFD_{avg} (in terms of LCC), and if the necessary increment in testing activity is compensated.

Consider in addition the impact of CCF on both the system performance measures and costs. Implementing redundancy implies that CCF is introduced, which is another factor that has an impact on the system's three main objectives: PFD_{avg} , STR and indirectly LCC. The PDS Method (Hauge et al., 2006a) shows that, for a fixed number of components N , as the M voting components increase the configuration factor C_{MooN} (Eq. (4.5)) increases as well (i.e. $C_{1oo3} < C_{2oo3}$, $C_{1oo4} < C_{2oo4} < C_{3oo4}$). This change is in the opposite direction for STR. Therefore, the impact of introducing voting into system optimization becomes less clear, and this complements the opportunities for research.

6.1.2. Modelling of MooN architectures

Modelling of k -out-of- n systems has several different approaches. The most basic approach is to evaluate their reliability using the binomial distribution $(S+F)^N$, where S =successful and F =Failed units, provided that all the components are identical (Billington & Allan, 1983). For reliability the distribution becomes $(r+q)^N$, being r =reliability and q =unreliability for each component. This is used in the general expression previously given in Eq. (1.12).

Biernat (1990) developed further this concept and obtained equations for reliability and MTBF of NMR systems (k -out-of- n :G where $n=2k-1$). He later developed algebraic models for estimation of the reliability bounds for systems with compensation of logical faults (Biernat, 1994, 1995). Moustafa derived (from continuous Markov chains) simultaneous linear differential equations for estimation of reliability of k -out-of- n :G systems with two failure modes with and without repair (Moustafa, 1996), and systems with dependent failures and imperfect coverage (Moustafa, 1997). Dudit & Rauzy (2001) analyzed the case of k -within- r -out-of- n systems (where r is a window of consecutive components in a linear arrangement) with solution by BDD. Arulmozhi (2002) proposed a procedure to compute exact reliability of k -out-

of- n : G systems with a recursive algorithm that generates all possible failure/no-failure combinations.

Due to the higher level of complexity, the approach for analysis of SIS has been rather based on combinatorial methods, a review of which was given in Chapter 1. The brief account presented here is focused on MooN architectures. Frederickson (1990) and Frederickson & Beckman (1991) used MA for determination of MTBF of dual and triple PLCs. Bodsberg & Hokstad (1995, 1997) presented the VULCAN model, a method based on RBD. Gruhn (1996) addressed in practical SIS the problem of choosing the best combination of architectures and components for design of a SIS. He included diagnostic coverage, CCF rates, and manual TIs. His approach is to use MA for the logic solver and FTA for the whole system.

IEC 61508-6 suggests a methodology for evaluation of PFD_{avg} based on simplified equations obtained from RBDs. The PDS method (Hauge et al., 2006a) is more sophisticated, but is also based on SE. A significant contribution of the method is the introduction of the multiple β factor model (developed in Hokstad & Corneliusen (2004), Hokstad (2004) and Hokstad et al. (2006)) and the modification factor C_{MooN} that alters the value of the β factor according to the different MooN architectures (Eq. (4.5)). Goble (1998) developed models for several voting configurations using both FTA and MA, including dangerous and safe failure modes, CCF and diagnostic coverage. He incorporated into the models the effect of the diagnostic circuit into the voting outcome in MooND architectures. Wilton (1998) used MA (with some simplifications) to determine MTTF and the Risk Reduction Factor (RRF) of 1oo2D and TMR architectures. ISA TR84.0.02 (ISA, 1999) intends to provide a standard set of methods for evaluation of SIL, using SE, FTA or MA.

Recently, Lu & Lewis (2006) presented formulas for evaluation of unavailability and probability of spurious operation of k -out-of- n systems ($Q_{k/n}$, $S_{k/n}$) based on the binomial distribution for independent failures. In addition, they developed equations for CCF based on the load-strength interference model. Analytical comparisons were made between two types of Emergency Shutdown Systems (ESD) in NPPs with 2oo3 and 2oo4 configurations. The study was subsequently extended to any k -out-of- n configuration (Lu & Lewis, 2008), although only embracing independent failures, with a relevant analysis of the sensitivity of $Q_{k/n}$ and $S_{k/n}$ to changes in n and k . Lu & Jiang (2007) made a comparison of impact that three on-line test and maintenance strategies (corrective, time-base preventive and condition-based predictive maintenance) have on the $Q_{k/n}$, $S_{k/n}$ and operation and risk costs of k -out-of- n systems (considering only independent failures).

6.1.3. Optimization with MooN systems

Ben-dov (1980) gave an early example of reliability optimization of k -out-of- n systems subject to two types of failure, with quantification by analytical methods seeking to find the optimal k given n . Bai et al. (1991) studied optimization of k -out-of- n systems with CCF, intending to find the optimal n for different combinations of k based on the ratios CCF/independent-failure and repair/acquisition costs. Pham & Galyean (1992) approached minimization of cost by finding the optimal number of spares S in a NFSR system. Pham (1992) included acquisition plus failure cost when finding the optimal k given a fixed n . Pham (1993) then addressed the search for optimal number of spares for a TMR+S system for optimal reliability, considering diagnostic coverage and CCF. Chari (1994) looked for the optimal k -out-of- n redundancy under two kinds of CCF (lethal and non-lethal) for cost minimization. Amari et al. (2004) incorporated imperfect fault coverage in the reliability and cost optimization analysis

Suich & Patterson (1991) sought to minimize the total and failure (average loss) costs of a k -out-of- n :G system by both selection of n and n/k combinations. This is an early analysis of building redundancy into a subsystem. Subsequently, Coit & Liu (2000) extended the analysis to series systems with k -out-of- n subsystems that could be either active or standby ones. The reliability optimization was executed by integer programming, subject to cost and weight constraints. This was later extended (Coit, 2003) to include imperfect sensing and switching in optimization of system reliability with component selection and redundancy allocation and choice of redundancy strategy of either active or cold-standby redundancy (integer programming).

Optimization of k -out-on- n architectures with genetic algorithms (GA) has been included in a few studies. Coit & Smith (1996a) used a single objective GA for redundancy and component selection: reliability maximization subject to cost and weight constraints. The problem is optimization of systems where k -out-of- n :G subsystems can be chosen. They are solved by the binomial distribution formula. In a subsequent work (Coit & Smith, 1996c) they estimated the quantification of reliability of a large system with k -out-of- n redundancies using neural networks. Pattison & Andrews (1999) implemented the SO GA optimization of a HIPS with component selection and where the number n of transmitters per subsystem and the number k necessary to trip were decision variables. The system dependability is estimated by using fault trees synthesized by BDD. The single-objective optimization is for system unavailability with cost, Spurious Trip Rate and maintenance time constraints. The methodology was scaled up for the larger problem of the firewater deluge system (Andrews & Bartlett, 2003). The work of the last two references was later adapted to multi-objective optimization in Borisevic & Bartlett (2007a, 2007b) and Riauke & Bartlett (2008).

Cantoni et al. (2000) combined a single objective GA optimization with Monte Carlo simulation design of a series-parallel system with k -out-of- n :G and cold-standby architecture options (redundancy and components selection), later extended to multi-objective optimization (Marseguerra et al., 2004b). A nuclear reactor protective system is analyzed, which contains a 2oo4 system analyzed by MA. The optimization is not for plant design but for test and maintenance policies. Tavakkoli et al. (2008) proposed single-objective reliability optimization by GA, where the redundancy strategy of each individual subsystem can be selected between active and cold k -out-of- n standby schemes.

6.2. MODELLING PFD FOR MooN ARCHITECTURES

The objective of this section is to present an analysis to extend the PFD(t) model (presented in the Chapter 5) for application to MooN architectures. This model has the power to explicitly model CCF, diagnostic coverage and changes in the mean test cycle of the components. This makes it convenient for optimization of design and testing policies. In contrast, quantification of dependability based on the binomial distribution models (Section 6.1.2) is limited to independent and identical components. Other studies that have approached it at system level either ignore the level of modelling required for quantification of PFD according to IEC 61508 requirements or result in very complex models difficult to handle in optimization cases.

6.2.1. Previous considerations

Safety Instrumented Systems are generally composed of sensors, logic solvers and actuators. Some authors have chosen to use “switch” diagrams to demonstrate the failure mechanisms of the logic solver (Goble, 1998) and the trip logic of protective systems (Lu & Lewis, 2008). This is applicable for showing the logic followed by MooN voting configurations for transmitters and, under some assumptions, actuators.

This study assumes that SIS are configured as de-energize-to-trip systems. This means that for execution of the safety function the system de-energizes the components. In addition, components are usually open-to-trip, which means a switch will open to execute the safety action. Therefore, for a MooN architecture to execute the safety function, M channels (or components) must agree to de-energize the system output. It is also considered that a parallel system is a particular case on MooN, where $M=1$. Finally, it is assumed that the voter is a perfect mechanism (i.e. reliability=1). This a valid assumption in SIS because the voter is usually a very simple mechanism, with very low failure rates (in comparison with the voting components themselves) and not subject to CCF (when it is not redundant, as is the most frequent case).

Consider the six different states that a component can have, showed in Figure 6.1. The first row corresponds to normal states of a component. The closed (energized) state corresponds to the component in service, ready to execute the safety action: to trip the system opening the switch. The second row corresponds to failure states. Notice that the dangerous failure is when the component fails stuck-closed, and the safe failure is stuck-open.

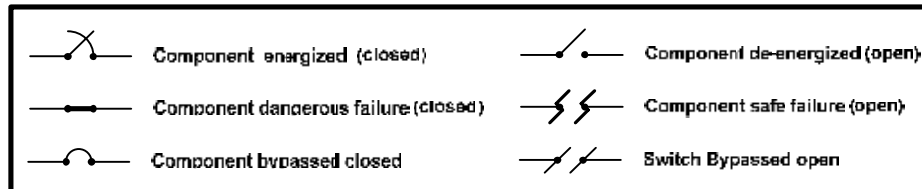


Figure 6.1. Symbols for the different states of a component

The third row in Figure 6.1 shows possible states for a component being under test. During test of one of its components the system is reconfigured, which is similar to the system being degraded during failure of one component. This means that the MooN architecture assumes a different MooN arrangement during test. The implementation of the physical reconfiguration in the system is made through bypasses. Whether it is bypassed open or closed depends on the specific operational philosophy. The PDS Method (Hauge et al., 2006a) discusses three basic philosophies:

1. Always shutdown during testing
2. Degraded operation during testing, otherwise shutdown,
3. Continue production during testing, even with no protection.

The operational philosophy can either be selected from these three or made from a combination of them. This determines how the system is reconfigured during test. For example, a 2oo3 system may become either a 2oo2 or a 1oo2 during test (dependant on how it is bypassed). This is very relevant because it shapes the CCF contribution to the PFD(t) during test, which is analyzed later.

6.2.2. Bypassing philosophy during test

Consider the analysis of a 1oo2 voting system shown in Figure 6.2. The figure presents the voting logic, how many faulty components can fail the system (dependant on the failure mode), and the result of bypassing one component (under test) by the two possibilities: open and closed bypass. Since $N=1$, this means that one single component is needed to vote for execution of the safety function (de-energize the load). Notice that if the bypass (for test) is open, the system would have to be taken out of service (plant shutdown), which in some way would negate the benefits of redundancy implemented to ensure uninterrupted operation. In contrast, bypassing

one component closed reconfigures the system to 1oo1 during test, which allows continuity of operation.

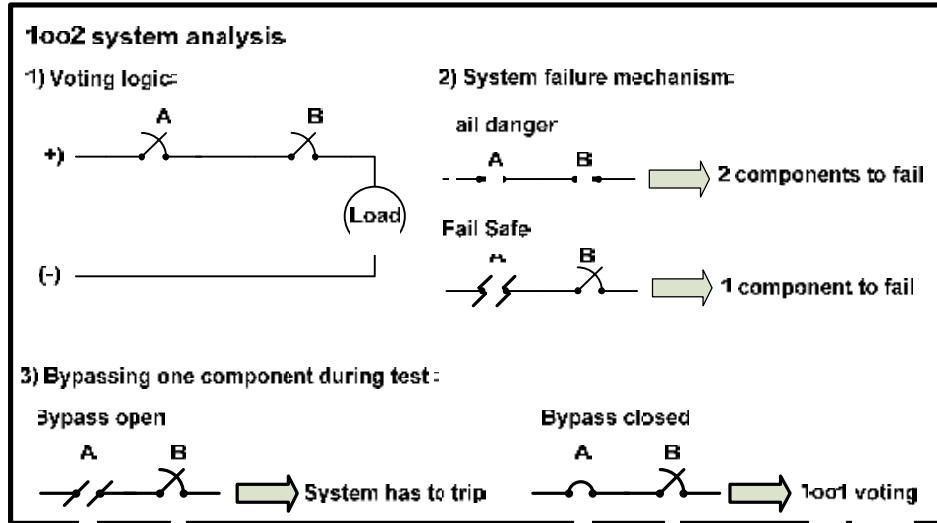


Figure 6.2. Analysis of 1oo2 architecture

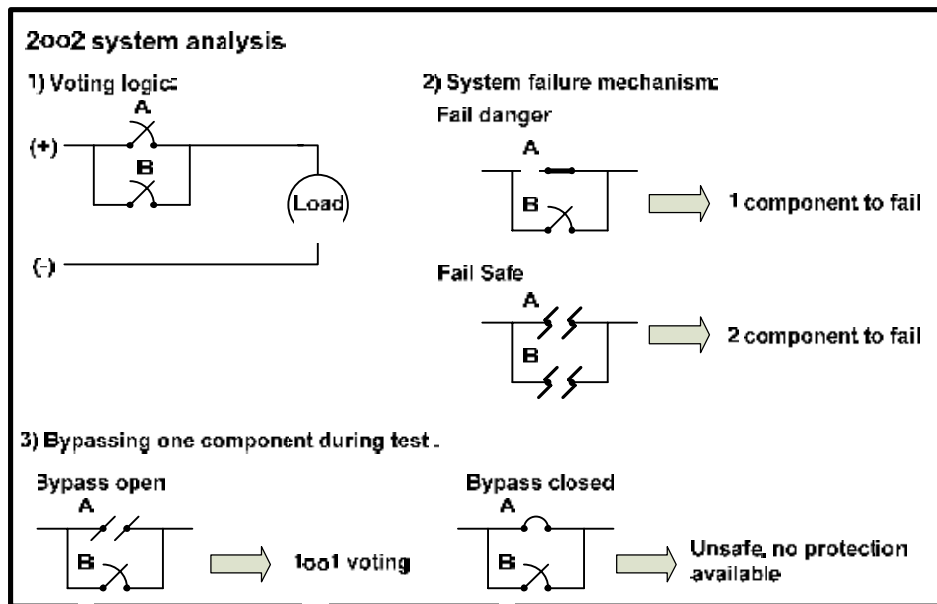


Figure 6.3. Analysis of 2oo2 architecture

A similar analysis for the 2oo2 architecture is shown in Figure 6.3. Observe the bypassing during test. If one component is bypassed closed, this would make the system unsafe since it would not actuate upon a demand during test whatsoever. This would make the system unavailable. In contrast, bypassing the component open would reconfigure the system to 1oo1

voting, permitting the system to keep performing its intended function. Thus, this is the more convenient bypassing during test.

It is evident that, as a result of this analysis, the selection of bypasses for both architectures, 1oo2 and 2oo2, must be different. For the 1oo2 architecture the bypass is left open so to avoid a plant shut down. For 2oo2 the bypass is closed, so that the system still performs its intended function during test of one component. Since the system is made of only two components, both architectures, 1oo2 and 2oo2, reconfigure to a 1oo1 arrangement during test (which is different for MooN systems where $N > 2$, as it will be seen later).

Relating this bypassing strategy to the operation philosophies enumerated in Section 6.2.1, it is clear that it would correspond to a combination of philosophies 2 and 3. This philosophy can be described as “continue production during testing with degraded operation in the safest possible way”. This means that for 1ooN systems, the component during test would be bypassed closed to permit continuous production, reconfiguring the system to a 1oo(N-1) configuration. In a different fashion, for MooN architectures with $M > 1$ the bypass would be open, permitting the system to keep performing its intended function in the safest possible way (i.e. with lowest PFD_{avg} possible). This is verified for Moo3 architectures in Figure 6.4.

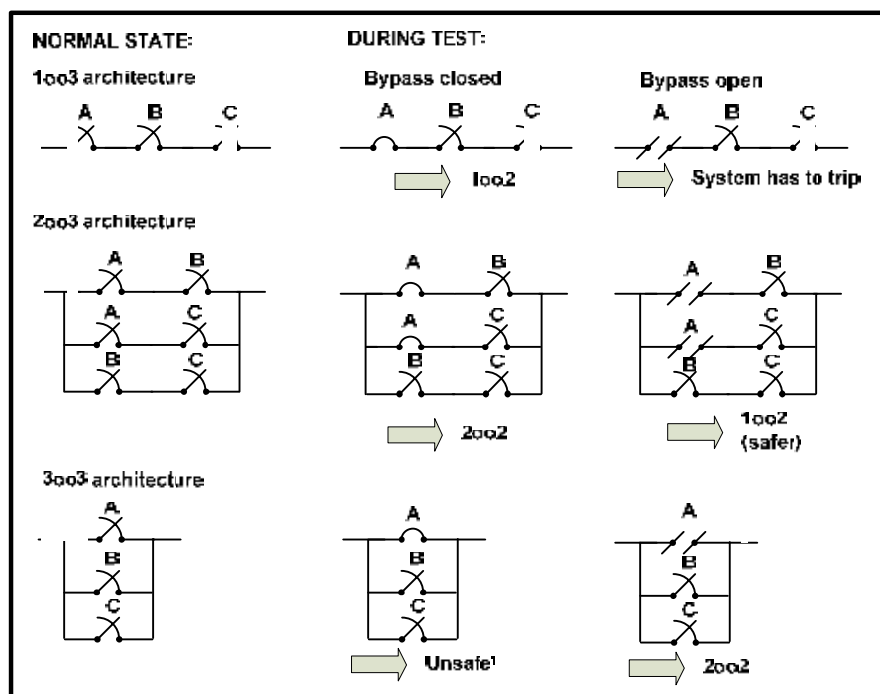


Figure 6.4. Analysis of bypasses for Moo3 architectures

Observe that for the 1oo3 architecture an open bypass would result in a system trip. On the contrary, a closed bypass allows reconfiguration to a 1oo2 arrangement, keeping the safety function in place. For the 2oo3 configuration, the close bypass reconfigures the system to 2oo2, while the open bypass reconfigures to 1oo2. Since the $PFD_{1oo2} < PFD_{2oo2}$, the open bypass is considered safer. For the 3oo3 system a closed bypass is unsafe since it is equivalent to bypassing the entire system. Observe that this bypass would annul the action of any of the other two working components. Therefore, the open bypass, which reconfigures to 2oo2 is safer. This pattern is practically the same for any MooN system where $N > 3$. Therefore, the bypassing philosophy to be used is:

- Bypass closed for 1ooN architectures.
- Bypass open for MooN architectures where $M > 1$.

6.2.3. Reduction of fault trees

Different MooN architectures keep a different balance between PFD and STR. This is the main reason for their application in safety system design. A particular architecture results in either dangerous or safe system failure as consequence of different combinations of the number of faulty components. This is modelled in the fault trees of the system. Fault trees for MooN systems where $N > 2$ can become very big and unmanageable. Fortunately, it is possible to reduce its size by Boolean algebra working with their logical expressions. Take the example of the 2oo3 architecture of Figure 6.5. It can be seen how the fault tree is reduced. This mechanism helps to simplify a fault tree with many possible combinations of repeated failure basic events.

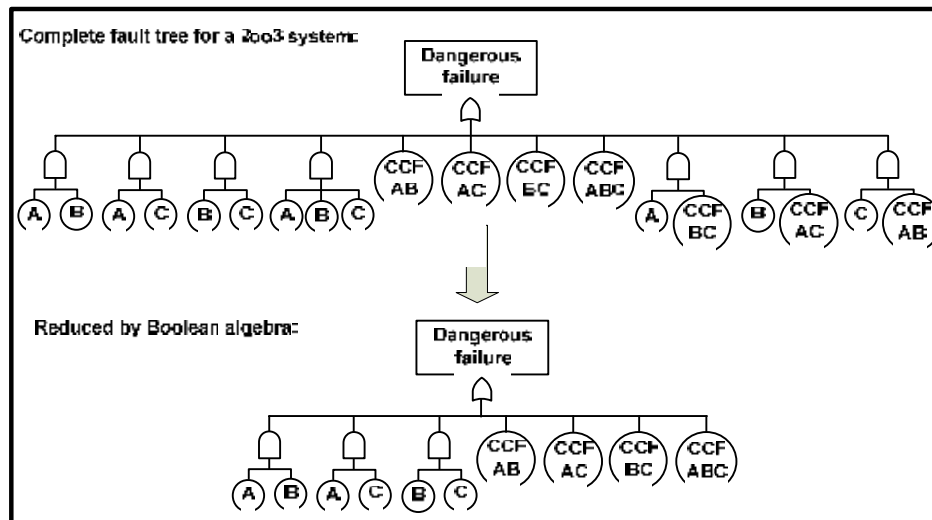


Figure 6.5. Example of fault tree reduction of a 2oo3 architecture

One important thing to notice here is that the CCF events of two components cannot be further reduced. Notice that in this example they are the most influential factors on the total probability of failure given that they are first order cut sets. For safety systems, probabilities of failure are very small, which means that multiplying two or more basic events (cut sets of order higher than one) gives events with an insignificant probability of failure compared with the first order events. This can be very helpful when further reducing the fault tree to get an acceptable approximation of the total probability of failure. This is the approach used here when treating fault trees.

6.2.4. CCF during test

When a component of a MooN architecture ($N > 1$) is bypassed during test its independent-mode failure is irrelevant because it would not affect the system's performance. Nevertheless, a component under test is not in a failed state and, therefore, it still can fail. Therefore, a CCF would still affect the system's functioning since it could lead to a simultaneous failure of the other components of the redundant arrangement. For instance, in a Moo2 system, a (common cause) failure of the component under test can lead the component not being tested to fail, and then to fail the entire arrangement. This means that putting the CCF contribution to zero (during test) in the architecture's fault tree would not be accurate.

An example serves to illustrate the concept: A system with two redundant transmitters dependant on a single power supply. One of the transmitters is taken out of service for test, being bypassed. At this time the second transmitter is still working and susceptible to failure. If the power supply fails at that moment both components will fail, the one being tested and the one being not. Thus, the system will fail. Notice that the bypass of the component under test does not protect the other transmitter from the CCF. This is a simple example, but the same situation would occur if we consider another different potential cause of CCF; e.g. subcomponents from the same manufacturer used in both transmitters, poor training in the person installing them, poor high-frequency isolation of sensitive equipment, etc.

The effect can be seen in Figure 6.6. Observe the first figure indicating the separation of events (failure modes). In the figure, $Q(A)$ and $Q(B)$ represent only the independent fractions. For a system 2oo2, the probability of failure is $Q(f) = Q(A) + Q(B) + Q(\text{CCF})$. However, during test this would become $Q(f) = Q(A) + Q(\text{CCF})$, since CCF is still relevant. Notice that this is equal to the total failure rate of the component, which is congruent for a single component.

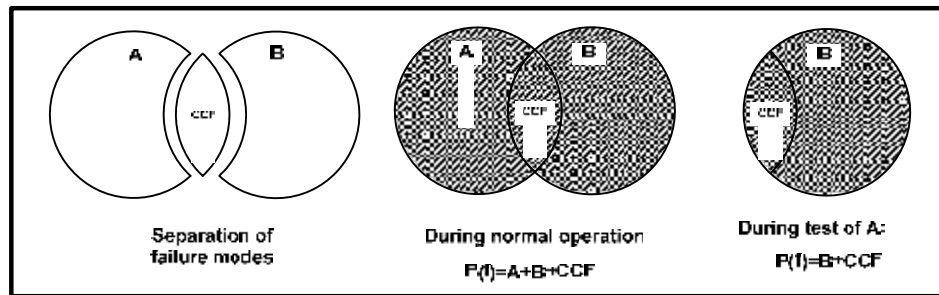


Figure 6.6. Probability of failure for a 2oo2 architecture

6.2.5. Reconfiguration of fault trees with bypasses

The reconfiguration of the safety system during test can be reflected by changing the fault tree of the system considering the values that the PFD corresponding to the independent components and CCF take during test. Observe the symbols for bypasses and for failure states in Figure 6.1. Although they are not the same, an open bypass resembles a safe failure (which impacts spurious trips), and a closed bypass resembles a dangerous failure (which impacts PFD). It could be then possible to consider the effects of bypasses (which reconfigure the system only during test) similar to failure effects in terms of PFD. For the 1oo2 architecture (Fig. 6.2) the bypass is closed (like a dangerous failure), and we could consider that the PFD of the component bypassed is equivalent to 1. Recalling that PFD is a measure of unavailability this concept acquires more congruence. On the contrary, if the bypass in the 2oo2 system is open (Fig. 6.3), it is like a safe failure, and then the PFD of the bypassed component could be considered equivalent to zero. In conclusion, independent failure events in fault trees would take the following values when their corresponding component is bypassed: bypass closed, event=1; bypass open, event=0. The concept is applied to the Moo2 architectures in Figure 6.7.

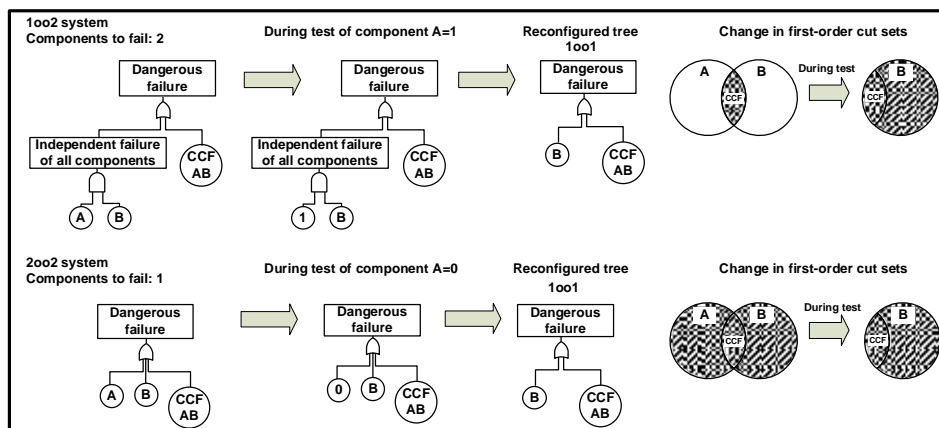


Figure 6.7. Reconfiguration of PFD fault trees of Moo2 architectures

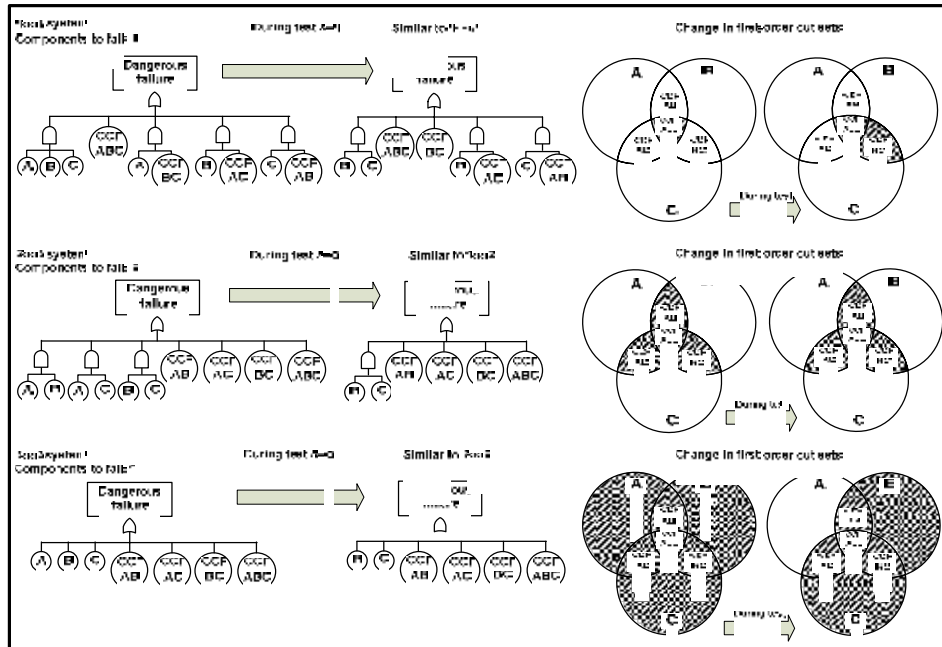


Figure 6.8. Reconfiguration of Moo3 architectures

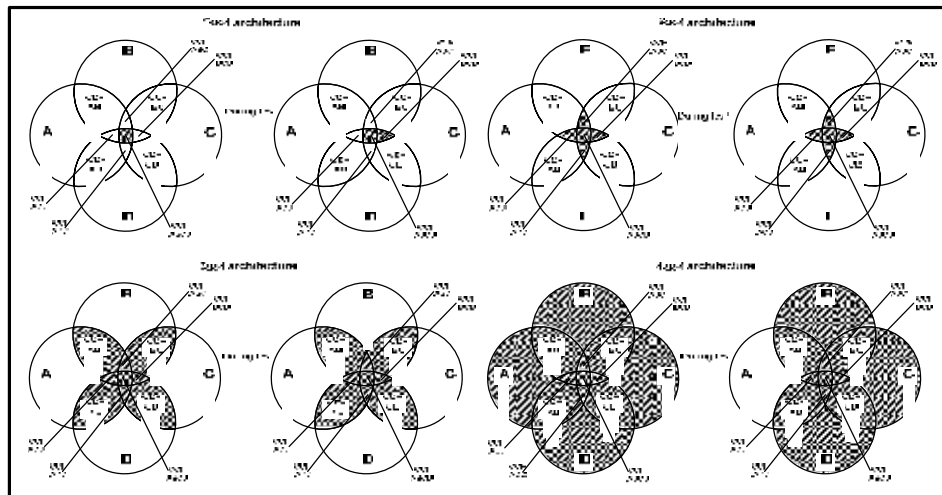


Figure 6.9. Reconfiguration of Moo4 architectures

Figure 6.7 shows how the fault trees of Moo2 architectures are reconfigured according to the type of bypass applied to one component under test. The last column shows the changes in PFD based on first-order cut sets. A similar analysis for Moo3 architectures is presented in Figure 6.8. Observe again the effect from the difference of bypasses: closed for 1oo3 and open for 2oo3 and 3oo3. Focus attention on the basic events that are directly related to the top event (are first-order cut sets). The 1oo3 tree changes to make the $Q(CCF_{BC})$ event a first order cut set during

test, which would increase its total probability of failure (PFD). Observe in contrast that for the 2oo3 system the first-order cut sets do not change; they are the same sum of all the Q(CCF) events. Since during test the reconfiguration only affects second-order cut sets, the total PFD practically does not change during test. This is assuming that the system does have some degree of CCF. If the system did not have CCF, the PFD would actually decrease during test, since two of its three (second-order) cut sets would be eliminated during test. Finally, observe that during the reconfiguration for the 3oo3 system the cut set of the independent failure of the component under test (A) is eliminated, which actually reduces the PFD during test.

Also notice that the reconfiguration of each MooN system during test has a similarity with another different $M^{\circ}ooN^{\circ}$ system (plus the CCF basic events). The 1oo3 system is practically reconfigured into a 1oo2, while 2oo3 and 3oo3 are changed to a 1oo2 and 3oo3 configuration respectively. A similar situation has been seen for the 1oo2 and 2oo2, which both change to a 1oo1 system. For architectures with $N > 3$ the pattern repeats itself (Fig. 6.9 demonstrates for Moo4 architectures). It can be concluded that during test 1ooN architectures are reconfigured to 1oo(N-1), while MooN (with $M > 1$) are changed to (M-1)oo(N-1) architectures. It is evident that the difference between having CCF and designing it out of the system has a significant influence on the system probability of failure. Table 6.1 summarizes the analysis for MooN architectures with $N \leq 4$, and presents on the bottom row a generalization for any MooN arrangement. It also indicates the number of faulty components able to fail the system in both dangerous and safe mode.

It is important to notice that the Venn Diagrams represent the probability of system failure based on only first-order cut sets. Also, notice that the intersections are denominated as CCF of certain combinations of components in the diagrams (for example CCF ABC) for simplicity. This represents the probability of simultaneous failure by CCF of the denominated components (A, B y C in this example).

Table 6.1. Analysis of MooN systems

N	MooN	Components to fail dangerous	Components to fail safe	Bypass	Reconfiguration during test
2	1oo2	2	1	closed	1oo1
	2oo2	1	2	open	1oo1
3	1oo3	3	1	closed	1oo2
	2oo3	2	2	open	1oo2
	3oo3	1	3	open	2oo2
4	1oo4	4	1	closed	1oo3
	2oo4	3	2	open	1oo3
	3oo4	2	3	open	2oo3
	4oo4	1	4	open	3oo3
N	1ooN MooN	N-M+1	M	closed open	1oo(N-1) (M-1)oo(N-1)

6.2.6. Estimation of PFD based on first-order cut sets

The Venn diagrams shown in Figures 6.7-6.9 represent the PFD based on first-order cut sets. Analyzing the diagrams, it is possible to identify the pattern that the PFD follows for any MooN architecture:

1. For 1ooN systems the PFD is approximately equivalent to the probability of CCF of N components $Q(\text{CCF}_N)$, and during test it is equivalent to the probability of CCF of N-1 components $Q(\text{CCF}_{N-1})$ (for $N \geq 3$).
2. For MooN systems where $1 < M < N$ the PFD is approximately equivalent to the contribution of given by probabilistic sum (union) of $\binom{N}{N-M+1}$ non-mutually CCF events of N-M+1 components; i.e. $\binom{N}{N-M+1} \cdot \text{CCF}_{N-M+1}$ (notice that N-M+1 is the number of faulty components necessary for the system to fail dangerously, see Table 6.1.). During test this equivalence remains the same.
3. For NooN systems the PFD is approximately equivalent to the sum of the probability of N independent failures plus the PFD equivalent to $\binom{N}{N-M+1} \cdot \text{CCF}_{N-M+1}$. During test this equivalence only changes in the contribution of the number of independent failures to N-1.

It can be noticed that the PFD can be quantified separating the independent and CCF components and applying them in combinations relevant to the first-order cut sets. This would permit their use in the PFD(t) model. The PFD of 1ooN and MooN architectures (point 1 and 2 above) requires only quantification of the shares related to different combinations of CCF events. These can be quantified determining the different CCF rates by modifying the β factor choosing the relevant C_{MooN} factors (Eq. (4.5)). For the NooN architectures it is actually necessary, in addition, to determine the share of independent failures (given that they are first-order cut sets) so they can be used in the PFD(t) model.

As described in Section 4.1.2, the PDS method (Hauge et al., 2006a) proposed the use of a C_{MooN} modification factor that changes the value of the β factor according to the specific MooN voting logic (Eq. (4.5)). Thus, the CCF rate of a single component is determined by substituting Eq. (4.5) into Eq. (4.4):

$$\lambda^C = \beta' \lambda^T = \beta \cdot C_{\text{MooN}} \cdot \lambda^T \quad (6.1)$$

The PDS method provides a formula for calculating the C_{MooN} factors. It is based on the multiple β factor model (Hokstad & Corneliussen, 2004; Hokstad, 2004). It assigns a value to the multiple betas based upon expert judgment, and then calculates the C_{MooN} factor. These formulas are not relevant here, and are thus omitted. The PDS method, however, gives other equations relevant to this study (reproduced in Eqs. (6.2-6.5)). The expression for all failures in a system (i.e. single, double, triple, etc), is given by Eq. (6.2). This can be considered the total rate of a NooN system:

$$\lambda_{\text{NooN}}^{\text{SYS}} = (N - C_N \cdot \beta) \cdot \lambda^T \quad (6.2)$$

Where λ^{SYS} is the total failure rate of the system, and λ^T the total failure rate of one single component. In addition:

$$C_N = \sum_{M=1}^{N-1} C_{\text{MooN}} \quad (6.3)$$

The independent failure rate of one single component (λ^N) in a MooN system is determined by:

$$\lambda^N = (1 - H_N \beta) \lambda^T \quad (6.4)$$

Where:

$$H_N = \frac{C_N + C_{(N-1)\text{ooN}}}{N} \quad (6.5)$$

Notice that the β factor used in Eqs (6.2) and (6.4) is the basic β used for two components. Table 6.2 reproduces the table given by the PDS method for MooN architectures up to N=6.

Table 6.2. Values of C_{MooN} , C_N and H_N (Hauge et al., 2006a)

	C_{MooN}					C_N	H_N
	M=1	M=2	M=3	M=4	M=5		
N=2	1.0	-	-	-	-	1.0	1.0
N=3	0.30	2.4	-	-	-	2.7	1.7
N=4	0.15	0.75	4.0	-	-	4.9	2.2
N=5	0.08	0.45	1.2	6.0	-	7.7	2.7
N=6	0.04	0.26	0.8	1.6	8.1	10.8	3.2

Eq. (6.2) can be manipulated to get the estimation of the system total failure rate considering the separation between independent failures and CCF rates of a NooN system.

$$\lambda_{NooN}^{SYS} = (N - C_N \beta) \lambda^T = (N - C_N \beta - C_{(N-1)ooN} \beta + C_{(N-1)ooN} \beta) \lambda^T \quad (6.6)$$

This can be rearranged to:

$$\lambda_{NooN}^{SYS} = \left(N \cdot \left[1 - \frac{C_N}{N} \beta - \frac{C_{(N-1)ooN}}{N} \beta \right] + C_{(N-1)ooN} \beta \right) \lambda^T = \left(N \cdot \left[1 - \left(\frac{C_N}{N} + \frac{C_{(N-1)ooN}}{N} \right) \beta \right] + C_{(N-1)ooN} \beta \right) \lambda^T \quad (6.7)$$

Substituting the equivalence by the term H_N :

$$\lambda_{NooN}^{SYS} = N(1 - H_N \beta) \lambda^T + C_{(N-1)ooN} \beta \lambda^T = N \lambda^N + \lambda_{(N-1)ooN}^C \quad (6.8)$$

This equation meets the definition of equivalence of PFD for NooN systems in Eq. (6.2). The equation indicates that the independent failure rate of one single component can be calculated using Eq. (6.4). Thus for a NooN system $\beta^i = \beta \cdot C_{(N-1)ooN}$.

Table 6.3 summarizes what has been exposed so far, where $Q(\text{CCF}_N)$ represents the probability of failure by CCF on N components, and $Q(\text{Ind})$ represents the probability of independent failure of one single component. Columns 3 and 4 show the modification factor C_{MooN} to be used for quantification of the overall CCF contribution, both during normal operation and during test.

Table 6.3. Quantification of PFD based on first-order cut sets

Architecture	PFD Normally	PFD During test	C_{MooN} Normally	C_{MooN} During test	PFD behaviour during test
1ooN	$Q(\text{CCF}_N)$	$Q(\text{CCF}_{N-1})$	C_{1ooN}	$C_{1oo(N-1)}$	Increases
MooN ($1 < M < N$)	$Q(\text{CCF}_{\text{MooN}})$	$Q(\text{CCF}_{\text{MooN}})$	C_{MooN}	C_{MooN}	No change
NooN	$N \cdot Q(\text{Ind})$ $+ Q(\text{CCF}_{(N-1)ooN})$	$(N-1) \cdot Q(\text{Ind})$ $+ Q(\text{CCF}_{(N-1)ooN})$	$C_{(N-1)ooN}$	$C_{(N-1)ooN}$	Decreases

6.2.7. Estimation of PFD considering null CCF

The PFD can be approximated very well considering only first-order cut sets (Table 6.3). This is possible because the contribution of higher-order cut sets is negligible compared with the first-order ones. However, according to Table 6.3, for 1ooN and MooN systems, if there was no CCF ($\beta=0$) the PFD would become zero. This is clearly unrealistic. It is therefore necessary to consider the independent failure contributions given that they would solely shape the PFD when CCF does not exist.

One solution would be to include all the cut sets of the fault tree, but this would lead to complications in the application of the PFD(t) model. Consider as an example the 1oo3 architecture (Fig. 6.8). It is evident that any basic event of multiple combinations of 1 independent failure plus CCF of two components has a much higher contribution to PFD than the triple independent failure of three components. Thus, it would be natural to include these combinations in the PFD quantification. This reasoning, however, would require to additionally estimate the $PFD_{CCF}(t)$ for three and for two components. This would be even more complicated for system with a larger number of components. In addition, the combinations of one (or more) independent failures with the CCF of two (or more) components are irrelevant when CCF exists, since the first-order cut sets dominate the PFD. Therefore, for cases where $CCF>0$ they are irrelevant in the PFD quantification. What is more, when $CCF=0$ these cut sets lose significance because they become zero. Therefore, in this case the dominant cut set in the 1oo3 system becomes the simultaneous independent failure of three independent failures. In general, for 1ooN system, this would be the simultaneous independent failure of the N components. It can then be concluded that when $CCF=0$ in 1ooN systems, only the combination of simultaneous independent failure of all the N components is to be considered.

Observe now the 2oo3 system (Fig. 6.8). There are three combinations of two independent failures. These combinations vary in MooN system for different M and N values. Regarding NooN architectures, however, the contribution of independent failure is always the sum of the independent failures of the N components, as already seen in Table 6.3. In addition, these combinations of independent failures change in the system fault tree when the system is reconfigured for test, which in turn is dependent upon the kind of bypass used. Based on fault trees (e.g. Figs. 6.7-6.8), a table can be developed to observe how these contributions change within a MooN system when M is increased, and how they change when the system is reconfigured. The results are presented in Table 6.4. The coefficients and exponents depend upon the number of faulty components necessary to fail dangerously the system, the formula for which was given in Table 6.1. Table 6.4 demonstrates that an identifiable pattern does exist, summarized in the last rows.

Table 6.4. General rule of contribution of independent failures to PFD

MooN	Contribution of independent failures	
	Normal state	During test
1oo2	$1 \cdot Q(\text{Ind})^2$	$1 \cdot Q(\text{Ind})^1$
2oo2	$2 \cdot Q(\text{Ind})^1$	$1 \cdot Q(\text{Ind})^1$
1oo3	$1 \cdot Q(\text{Ind})^3$	$1 \cdot Q(\text{Ind})^2$
2oo3	$3 \cdot Q(\text{Ind})^2$	$1 \cdot Q(\text{Ind})^2$
3oo3	$3 \cdot Q(\text{Ind})^1$	$2 \cdot Q(\text{Ind})^1$
1oo4	$1 \cdot Q(\text{Ind})^4$	$1 \cdot Q(\text{Ind})^3$
2oo4	$4 \cdot Q(\text{Ind})^3$	$1 \cdot Q(\text{Ind})^3$
3oo4	$6 \cdot Q(\text{Ind})^2$	$3 \cdot Q(\text{Ind})^2$
4oo4	$4 \cdot Q(\text{Ind})^1$	$3 \cdot Q(\text{Ind})^1$
1ooN	$Q(\text{Ind})^N$	$Q(\text{Ind})^{N-1}$
MooN (1>M>N)	$\binom{N}{N-M+1} \cdot Q(\text{Ind})^{N-M+1}$	$\binom{N-1}{N-M+1} \cdot Q(\text{Ind})^{N-M+1}$
NooN	$N \cdot Q(\text{Ind})$	$(N-1) \cdot Q(\text{Ind})$

6.2.8. Application to the PFD(t) model

As a consequence of the previous analysis, a formulation can be made establishing that the total time dependent PFD of a MooN architecture PFD_{MooN} can be determined by:

$$PFD_{\text{MooN}}(t) = PFD_{\text{IND}}(t) + PFD_{\text{CCF}}(t) \quad (6.9)$$

Where:

$PFD_{\text{MooN}}(t)$ = Total system PFD

$PFD_{\text{IND}}(t)$ = PFD by combination of independent failures

$PFD_{\text{CCF}}(t)$ = PFD by combination of CCF

The instant contributions PFD_{CCF} and PFD_{IND} were determined in Tables 6.3 and 6.4 respectively, and they are summarized in Table 6.5. The independent failure rates λ^N , to be used for calculating the $PFD_{\text{ind};i}(t)$, can be determined by Eq. (6.4). The $PFD_{\text{CCF}}(t)$ contribution can be estimated substituting the modification factors C_{MooN} determined in Table 6.5 into Eq. (6.1).

Table 6.5. Summary of formulae for quantification of PFD

Architecture	State	PFD _{IND}	PFD _{CCF}	C _{MooN}
1ooN	Normal operation	$Q(Ind)^N$	$Q(CCF_N)$	C_{1ooN}
	Test	$Q(Ind)^{N-1}$	$Q(CCF_{N-1})$	$C_{1oo(N-1)}$
MooN ($1 < M < N$)	Normal operation	$\binom{N}{N-M+1} \cdot Q(Ind)^{N-M+1}$	$Q(CCF_{MooN})$	C_{MooN}
	Test	$\binom{N-1}{N-M+1} \cdot Q(Ind)^{N-M+1}$	$Q(CCF_{MooN})$	C_{MooN}
NooN	Normal operation	$N \cdot Q(Ind)$	$Q(CCF_{(N-1)ooN})$	$C_{(N-1)ooN}$
	Test	$(N-1) \cdot Q(Ind)$	$Q(CCF_{(N-1)ooN})$	$C_{(N-1)ooN}$

The formulation presented in Table 6.5 can be used to determine how to adapt the PFD(t) model presented in Chapter 5 (Eqs. (5.10-5.18)) for modelling of MooN systems. Changes to the PFD(t) model itself are minimal, and the changes are more about the way CCF and independent failure rates are quantified. Secondly, Table 6.5 guides the construction of fault trees regarding independent failures. The contribution of CCF to PFD is still modelled as a single component, and what changes is the way the CCF rates are quantified using distinct C_{MooN} modification factors. Although the modifications to the model are minimal, it is included here in full to help the reader (Eqs. (6.10-6.19)).

PFD(t) model for MooN architectures. Independent failures contribution	
Standby before first test:	(6.10)
$PFD_{ind_i}(t) = \lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN} t}$ for $t - T_{Pi} < 0$	
Testing:	(6.11)
$PFD_{ind_i}(t) = C_{test}$ for $\{t - T_{Pi} \geq 0 \ \& \ 0 \leq w_i < T_i\}$	
Repair:	(6.12)
$PFD_{ind_i}(t) = \begin{cases} \lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN} T_{Pi}} + (e^{-\lambda^{DUN} T_{Pi}} - \lambda^{DDN} T_r)(\lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN} (w_i - T_i)}) & \text{for } \{0 \leq t - T_{Pi} < TI \ \& \ T_i \leq w_i < T_i + T_r\} \text{ \textit{first repair}} \\ \lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN} (TI - T_i - T_r)} + (e^{-\lambda^{DUN} (TI - T_i - T_r)} - \lambda^{DDN} T_r)(\lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN} (w_i - T_i)}) & \text{for } \{t - T_{Pi} \geq TI \ \& \ T_i \leq w_i < T_i + T_r\} \text{ \textit{subsequent repairs}} \end{cases}$	
Standby between tests:	(6.13)
$PFD_{ind_i}(t) = \lambda^{DDN} T_r + 1 - e^{-\lambda^{DUN} (w_i - T_i - T_r)}$ for $\{t - T_{Pi} \geq 0 \ \& \ w_i \geq T_i + T_r\}$	
where:	(6.14)
$w_i = (t - T_{Pi}) \bmod TI$	

Failure modes:

$$\lambda^{DDN} = \varepsilon \cdot (1 - H_N \beta) \lambda^D \quad (6.20)$$

$$\lambda^{DUN} = (1 - \varepsilon) \cdot (1 - H_N \beta) \lambda^D \quad (6.21)$$

$$\lambda^{DDC} = \varepsilon \cdot \beta \cdot C_{MooN} \cdot \lambda^D \quad (6.22)$$

$$\lambda^{DUC} = (1 - \varepsilon) \cdot \beta \cdot C_{MooN} \cdot \lambda^D \quad (6.23)$$

$$\lambda_{test}^{DDC} = \varepsilon \cdot \beta \cdot C_{TMooN} \cdot \lambda^D \quad (6.24)$$

$$\lambda_{test}^{DUC} = (1 - \varepsilon) \cdot \beta \cdot C_{TMooN} \cdot \lambda^D \quad (6.25)$$

Table 6.6. Modification factors

Architecture	C_{Test}	C_{MooN}	C_{TMooN}
1ooN	1	C_{1ooN}	$C_{1oo(N-1)}$
MooN (1<M<N)	0	C_{MooN}	C_{MooN}
NooN	0	$C_{(N-1)ooN}$	$C_{(N-1)ooN}$

The changes and adaptations derived from Table 6.5 are described next:

1. The only change to $PFD_{ind\ i}(t)$ is in Eq. (6.11) (originally Eq. (5.11)) for the contribution during test. $PFD_{ind\ i}(t)$ during test is equal to a constant C_{test} that takes a value depending on how it is bypassed during test (closed=1, open=0), see Table 6.6. The rest of the $PFD_{ind\ i}(t)$ model remains unaltered.
2. Independent failure rates (λ^{DN}) are quantified using the factor $H_N \beta$ (rather than simply β), Eqs. (6.20-6.21).
3. CCF rates are quantified based on distinct modification factors (Eqs. (6.22-6.25)). There are two factors, C_{MooN} and C_{TMooN} , one for normal operation and one for test respectively. These change according to the MooN architecture being modelled as indicated in Table 6.6. Also notice that the CCF failure rates during test are indicated as λ_{test}^{DDC} and λ_{test}^{DUC} .
4. The only change in the $PFD_{CCF}(t)$ model is made for the notation of CCF rates during test in Eq. (6.16) (originally Eq. (5.16)). The rest of the $PFD_{CCF}(t)$ remains unaltered.
5. Table 6.5 (column headed P_{IND}) guides the construction of fault trees of a specific MooN architecture related to the contribution of the independent failures of each component as follows: The coefficients of each formula determine the number of combinations of independent failures. The exponents indicate the number of basic events (independent failures) to be included per combination.

6.2.9. Application example

In order to demonstrate the functionality of the model, a case study is formulated. The maximum and average PFD is evaluated for MooN systems with up to five components. The data of the problem is: $\lambda^T=1.9 \times 10^{-6}$; $\beta=10\%$; $\varepsilon=50\%$; $T_r=8$ hrs; $T_i=1$ hr, $TI=1$ year, uniform staggered test, mission time $T=10$ years. The results are shown Figure 6.10.

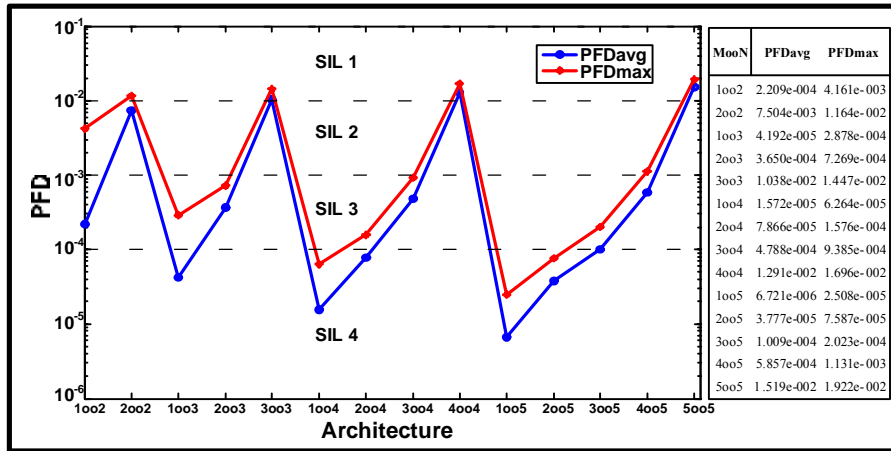


Figure 6.10. PFD of MooN architectures

The results obtained are congruent with what was formulated in Table 6.1 regarding the number of faulty components that can fail the entire system ($N-M+1$). As M increases (with fixed N) the number of faulty components that can fail the system is reduced, and thus the probability of dangerous failure increases, as can be seen in the graph. Also observe that the trend of the graph for $100N$ systems is congruent: as N raises (with $M=1$) PFD drops. Just for illustration purposes, Figure 6.11 shows the plot of PFD for a 4005 system. The plots show the independent and CCF (dotted red line) PFD(t) in the upper graph, while the lower graph illustrates the PFD_{max} (blue) and PFD_{avg} (black).

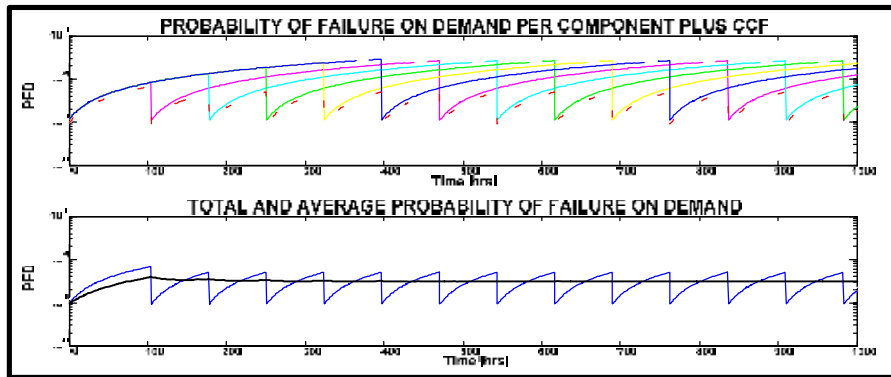


Figure 6.11. Plots of PFD(t) of a 4005 system

6.3. MODELLING OF STR FOR MooN ARCHITECTURES

6.3.1. Effects of bypass on the STR

This section is dedicated to study the behaviour of MooN during normal operation and their reconfiguration during test from the perspective of STR modelling. The analysis follows the same structure as for the one made for PFD modelling in Section 6.2. The bypassing philosophy established in that section is that the bypass will be closed for 1ooN (parallel) architectures closed and open for MooN ($M > 1$) architectures. It has been seen that the specific bypass for one component influences how the system fault tree is reconfigured during test. However, the effect that a bypass has on the spurious trips is different than the effect of a bypass on PFD. Actually, as with the PFD quantification, a bypass has an effect that resembles a failure, but in the opposite direction. Remember that spurious trips are safe failures. For PFD the bypasses behave in a similar way to a dangerous failure of the corresponding basic event in the fault tree. A closed bypass resembles a dangerous failure (and thus it takes a similar value in the PFD fault tree). However, regarding safe failure this would resemble the opposite, and thus a zero in the STR fault tree. This can be easily understood for the open bypass. This would resemble a no-failure for PFD (since the action to trip is an open switch). But for the STR this resembles a failure (because a false trip is always the result of an open state). Therefore, an open bypass corresponds to a value of one in the STR fault tree. In conclusion, the bypasses take the opposite value in the PFD and the STR fault trees. For STR a closed bypass is equivalent to a value of zero, and an open bypass a value of one.

6.3.2 Reconfiguration of STR fault trees with bypasses

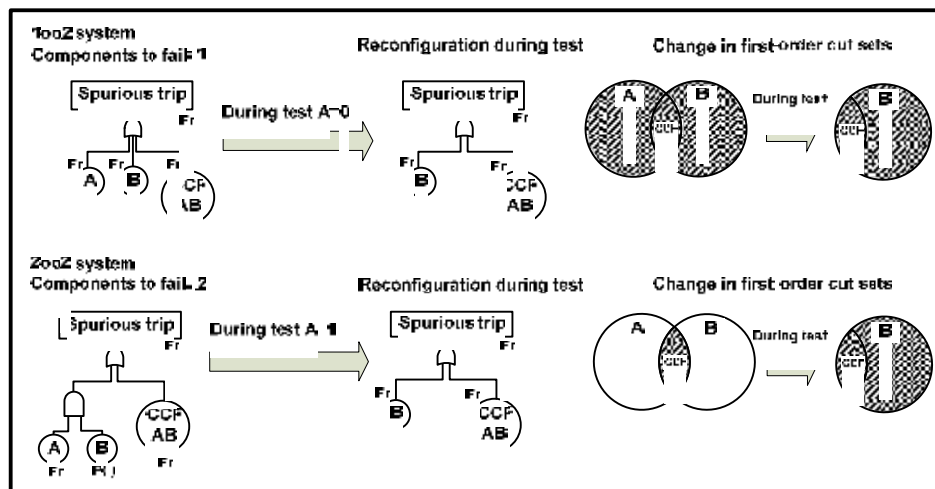


Figure 6.12. Reconfiguration of STR fault trees of Moo2 architectures

Figure 6.12 illustrates the reconfiguration of STR fault trees during test. It also shows the changes in the Venn diagrams related to the first-order cut sets. Observe the change of the value of component A during test according to the corresponding bypass: a closed bypass for the 1oo2 system makes the basic event $A=0$; an open bypass for the 2oo2 system makes the basic event $A=1$. As a consequence, according to the Venn diagram, the STR decreases for the 1oo2 system during test, while for the 2oo2 increases. It is important to remember that the Venn Diagrams represent the probability of system failure based only in first-order cut sets.

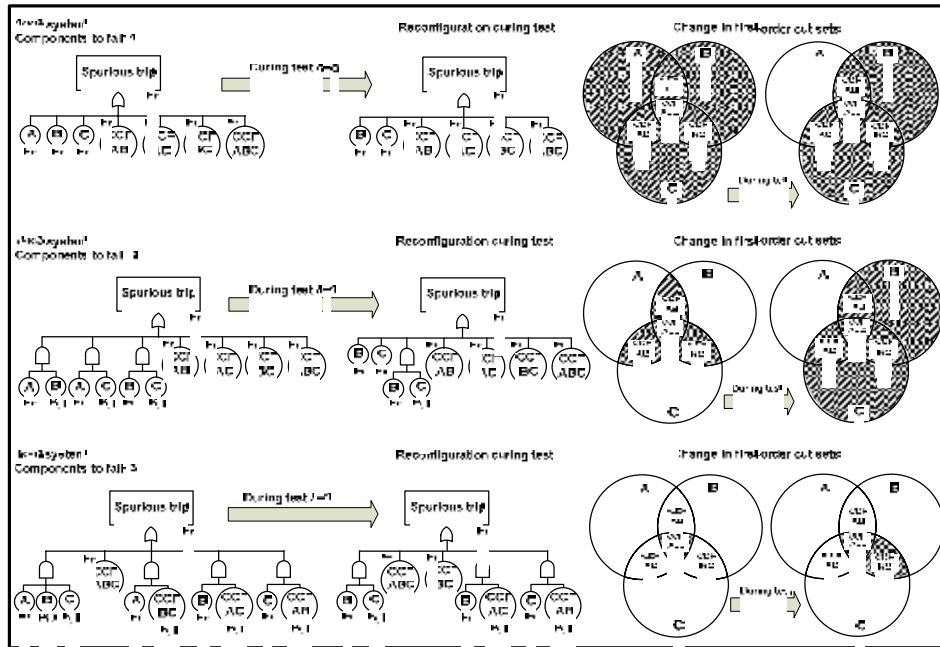


Figure 6.13. Reconfiguration of Moo3 architectures from the STR perspective

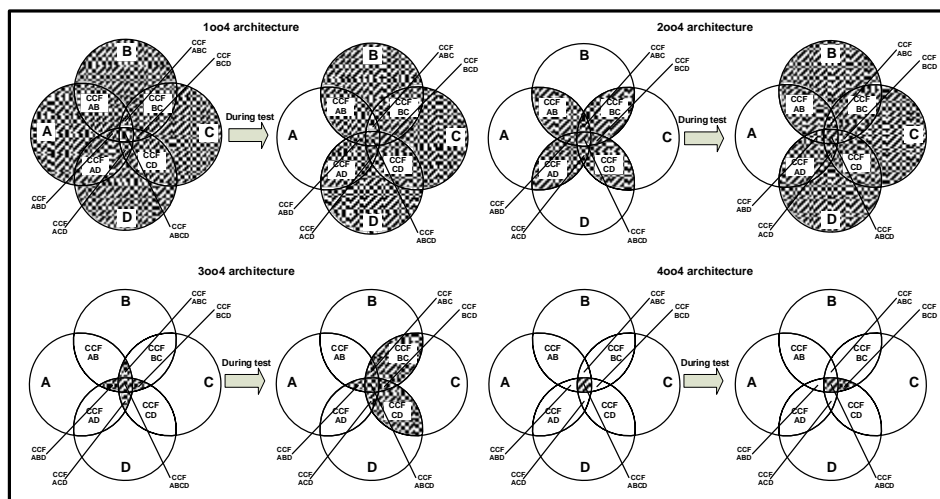


Figure 6.14. Reconfiguration of Moo4 architectures from the STR perspective

Different from the PFD, the STR is a measure of frequency (no probability). Frequency basic events cannot be multiplied in AND gates, which means that some of their basic events have to be converted to probabilities before quantifying the top event. This is indicated in the fault trees with “Fr” for frequency basic events, and with “P()” for probabilities. This problem did not arise during optimization of parallel systems because their fault trees contain only OR gates. However for systems with $M > 1$ this conversion has to be taken into account. Observe the AND gate in the fault tree of the 2oo2 system. The first basic event is the frequency of the failure of the A component, while the second one is the probability of the second component B. This similar situation will be seen for three-component systems in Figure 6.13. Notice again this change in the AND gates of the basic events. The same assumptions for the bypasses apply.

Figures 6.13-6.14 illustrate the changes in STR during test for architectures with three and four components based on the first-order cut sets. Observe these figures and compare them with Figures 6.8-6.9. It can be noticed that the CCF of the STR of a MooN architecture behaves in a similar fashion to the CCF of the PFD of a $(N-M+1)ooN$ architecture. The PDS Method (Hauge et al., 2006a) had previously arrived to a similar conclusion. It can be therefore formulated that the modification factor to use for estimating the CCF related to STR of a MooN system would be $C_{(N-M+1)ooN}$. Therefore, the CCF share can be calculated as:

$$\lambda^{SC} = \beta' \lambda^S = \beta \cdot C_{(N-M+1)ooN} \cdot \lambda^S \quad (6.26)$$

The safe independent failure rate of a single component (λ^{SN}) can be calculated based on Eq. (6.2), previously used for estimating dangerous independent failures:

$$\lambda^{SN} = (1 - H_N \beta) \lambda^S \quad (6.27)$$

In the same fashion as for PFD, and based on the conclusions drawn above, an equivalence of STR based on first-order cut sets can be established, which are shown in Table 6.7. In the table the probability of spurious failure related to CCF on N components is indicated as $\text{Psf}(\text{CCF}_N)$. The values indicated for the C_{MooN} modification factors can be taken from Table 6.2.

Table 6.7. Quantification of STR based on first-order cut sets

Architecture	STR	STR	C_{MooN}	C_{MooN}	STR behaviour during test
	Normally	During test	Normally	During test	
1ooN	$N\lambda^{SN}$ + $\text{Psf}(\text{CCF}_{N-M+1})$	$(N-1)\lambda^{SN}$ + $\text{Psf}(\text{CCF}_{N-M+1})$	$C_{(N-M)ooN}$	$C_{(N-M)ooN}$	Decreases
2ooN	$\text{Psf}(\text{CCF}_{N-M+1})$	$(N-1)\lambda^{SN}$ + $\text{Psf}(\text{CCF}_{N-M+1})$	$C_{(N-M+1)ooN}$	$C_{(N-M+1)ooN}$	Increases
MooN ($2 < M < N$)	$\text{Psf}(\text{CCF}_{N-M+1})$	$\text{Psf}(\text{CCF}_{N-M+1})$	$C_{(N-M+1)ooN}$	$C_{(N-M+1)oo(N-1)}$	Increases
NooN	$\text{Psf}(\text{CCF}_N)$	$\text{Psf}(\text{CCF}_{N-1})$	C_{1ooN}	$C_{1oo(N-1)}$	Increases

Several observations can be made from Table 6.7:

- The final values of the system STR are practically shaped by the CCF contribution in every MooN architecture.
- For 1ooN systems the STR decreases during test, while that for MooN architectures ($M > 1$) the STR decrease. This difference is caused by the kind of bypass used.
- 1ooN architectures have the N independent basic events as first-order cut sets, and thus these events have an important contribution in the STR.
- 2ooN architectures are different for other architectures (where $M > 2$): during normal operation only the CCF contribution is important, but during test the independent failures of N-1 components are added to the total STR as first-order cut sets.
- For MooN architectures where $M > 2$, only the CCF gives first order cut sets. At the same time, during test reconfiguration, the CCF contribution to the STR increases, and thus the modification factor C_{MooN} changes (which does not happen when $M \leq 2$). This can be observed comparing the Venn diagrams of Moo4 architectures for PFD and STR (Fig. 6.9 vs Fig. 6.14). For example, the Venn diagram of the 3oo4 architecture in the STR behaves like a 2oo4 system by PFD during normal operation (i.e. C_{1002}), but during test it behaves like a 2oo3 system (C_{2003}). Making a generalization, this means that the modification factor changes from $C_{(N-M+1)ooN}$ to $C_{(N-M+1)oo(N-1)}$. This generalization can be verified analyzing the logic of systems with larger M (with switch diagrams for example).

Table 6.8. General rule of contribution of independent failures to STR

MooN	Contribution of independent failures	
	Normal state	During test
1oo2	$2 \cdot \lambda^{SN}$	λ^{SN}
2oo2	$\lambda^{SN} \cdot \text{Psf}(SN)^1$	λ^{SN}
1oo3	$3 \cdot \lambda^{SN}$	$2 \cdot \lambda^{SN}$
2oo3	$3\lambda^{SN} \cdot \text{Psf}(SN)^1$	$2\lambda^{SN} + \lambda^{SN} \cdot \text{Psf}(SN)^1$
3oo3	$\lambda^{SN} \cdot \text{Psf}(SN)^2$	$\lambda^{SN} \cdot \text{Psf}(SN)^1$
1oo4	$4 \cdot \lambda^{SN}$	$3 \cdot \lambda^{SN}$
2oo4	$6\lambda^{SN} \cdot \text{Psf}(SN)^1$	$3\lambda^{SN} + 3\lambda^{SN} \cdot \text{Psf}(SN)^1$
3oo4	$4\lambda^{SN} \cdot \text{Psf}(SN)^2$	$3\lambda^{SN} \cdot \text{Psf}(SN)^1 + \lambda^{SN} \cdot \text{Psf}(SN)^2$
4oo4	$\lambda^{SN} \cdot \text{Psf}(SN)^3$	$\lambda^{SN} \cdot \text{Psf}(SN)^2$
1ooN	$N \cdot \lambda^{SN}$	$(N-1) \cdot \lambda^{SN}$
2ooN*	$\binom{N}{M} \cdot \lambda^{SN} \cdot \text{Psf}(SN)$	$\left[(N-1) + \left(\binom{N}{M} - (N-1) \right) \text{Psf}(SN) \right] \cdot \lambda^{SN}$
MooN (2>M>N)	$\binom{N}{M} \lambda^{SN} \cdot \text{Psf}(SN)^{M-1}$	$\left[\binom{N-1}{M-1} \text{Psf}(SN)^{M-2} + \left(\binom{N}{M} - \binom{N-1}{M-1} \right) \text{Psf}(SN)^{M-1} \right] \cdot \lambda^{SN}$
NooN	$\lambda^{SN} \cdot \text{Psf}(SN)^{N-1}$	$\lambda^{SN} \cdot \text{Psf}(SN)^{N-2}$

*The equations for 2ooN are a particular case derivation from MooN

Same as with the PFD, the total STR could be approximated based the contribution of the first-order cut sets given in Table 6.7. However, if the system did not have CCF the STR would be

equal to zero, which would be unrealistic. For this reason, the cut sets with order higher than one and with only independent failures in their basic events are also taken into account during the STR quantification. Table 6.8 shows the contribution by independent failures for MooN systems with up to $N \leq 4$. Here λ^{SN} is the safe independent failure rate, and $Psf(SN)$ indicates the probability of independent safe failure (here SN stands for Safe Normal failure mode). Following the contributions indicated in the table, it is possible to identify the pattern that is summarized in the equations presented in the last rows.

6.3.3. Probability of safe failures

Remember that the failure modes of a component are split by the automatic diagnostic coverage into detected and undetected (Eqs. (1.13-1.14)). Thus, the probability of safe failure is related to the failure residence time. For safe detected (SD) failures (assuming they are detected and repaired immediately) the probability of safe failure (Psf) is determined by the T_r (Eq. (6.28)).

In contrast, the residence time for safe undetected (SU) failures could be the entire system mission time T (or life time until an overhaul), see Eq. (6.30). This affects considerably the STR by internal failures (STR_i) of a redundant system. Observe the fault tree for a 2oo2 system (Fig. 6.12), and the formula for independent failures in Table 6.8. With two components voting in 2oo2, i.e. an AND gate, the failure rate of one component is multiplied times the probability of spurious (safe) failure of a second component, which may become quite large by the undetected failure mode.

In Chapter 5 it was considered that proof testing targeted only dangerous failures, and therefore it did not affect the STR_i . However, some authors consider that the proof test actually detects some safe failures (CCPS, 2007). If it is considered that proof test actually manages to detect some fraction of safe failures, an intermediate failure mode between detected and undetected would appear: the safe failures undetected by the automatic diagnostic coverage but detected by proof test (SUT). Thus, the probability of safe failures could be finally split into three failure modes:

$$Psf(SD) = \lambda^S \cdot \varepsilon \cdot T_r = \lambda^{SD} \cdot T_r \quad (6.28)$$

$$Psf(SUT) = \lambda^S \cdot (1 - \varepsilon) \cdot \varepsilon^{ts} \cdot (TI + T_r) = \lambda^{SU} \cdot \varepsilon^{ts} \cdot (TI + T_r) \quad (6.29)$$

$$Psf(SU) = \lambda^S \cdot (1 - \varepsilon) \cdot (1 - \varepsilon^{ts}) \cdot T = \lambda^{SU} \cdot (1 - \varepsilon^{ts}) \cdot T \quad (6.30)$$

Where:

ε^{ts} = Diagnostic coverage of proof test on safe failures

$Psf(SUT)$ = Safe undetected failures (by automatic diagnostics) but detected by the proof test

As mentioned above, if the proof test coverage ε^{IS} were 100%, Eq. (6.30) would become void. In contrast, if we consider that the proof test does not detect any safe failures ($\varepsilon^{IS}=0\%$), Eq. (6.29) would become void.

6.3.4. Quantification of STR modified during test

As seen in Figures 6.12-6.14, the total STR of a determined architecture changes during test as a consequence of the system being reconfigured by the bypass of the component under test. The STR of 1ooN architectures decreases during test, and for MooN ($M>1$) the STR actually increases during test and repair. The test-plus-repair time could seem that short (compared to the Test Interval) that it could be neglected. However, remember that this contribution is considered so important that it is being included in the quantification of PFD(t).

The question that arises is how to include the effect of the reconfiguration in the final STR if the model being used is not time dependent. Lu & Jiang (2007) estimated the relative time spent in each configuration (normal operation versus operation with one component under test). Then they multiplied the corresponding STR by the relative time, and summed up both contributions:

$$STR_{\lambda} = (Rt_N \cdot STR_N) + (Rt_{TR} \cdot STR_{TR}) \quad (6.31)$$

Where:

Rt_N = Relative time under normal operation

Rt_{TR} = Relative time under test and (possible) repair

STR_N = STR for normal operation

STR_{TR} = STR for test (and repair)

Substituting the value of relative times and reordering:

$$STR_{\lambda} = \frac{T - [T_t + T_r \cdot Q(f)] \cdot \frac{T}{TI}}{T} \cdot STR_N + \frac{[T_t + T_r \cdot Q(f)] \cdot \frac{T}{TI}}{T} \cdot STR_{TR} \quad (6.32)$$

Since:

$$Rt_{TR} = \frac{T_t + T_r \cdot Q(f)}{TI} \quad (6.33)$$

Therefore:

$$STR_{\lambda} = (1 - Rt_{TR}) \cdot STR_N + Rt_{TR} \cdot STR_{TR} \quad (6.34)$$

Where $Q(f)$ is the probability of the component under test being found failed would be given by Eq. (6.35), considering that $Q(f)=Q(t)=Q(t=TI)$:

$$Q(f) = q + 1 - e^{-\lambda TI} \quad (6.35)$$

Remember that q is equivalent to the unavailability as a result of (dangerous) detected failures in the PFD(t) model (Eqs. (6.10-6.19)). It is important to remember that in Chapter 5 it was assumed that proof testing detects only dangerous failures, so its benefits did not impact the STR quantification. This is not the case of its adverse effects. Therefore the test interval TI refers to the dangerous proof test. However, apart from test time, when a component is found dangerously failed it is kept out of service for the additional repair time. These test and repair times (of dangerous failures) thus affect the STR because during this time the total STR changes due to the temporary system reconfiguration.

Therefore, if it is considered that the proof test does detect some safe failures, the complete expression for Eq. (6.35), would become:

$$Q(f) = (\lambda^{DDN} + \lambda^{DDC} + \lambda^{SDN} + \lambda^{SDC})T_r + 1 - e^{-(\lambda^{DUN} + \lambda^{DUC} + \lambda^{SUTN} + \lambda^{SUTC})TI} \quad (6.36)$$

Notice that:

$$\lambda^{SUTN} = \lambda^{SUN} \cdot e^{-ts} \quad (6.37)$$

$$\lambda^{SUTC} = \lambda^{SUC} \cdot e^{-ts} \quad (6.38)$$

In Eqs. (6.36-6.38), the new failure rates nomenclature is:

λ^{SUTN} = Safe undetected by automatic diagnostics, but detected by test (independent)

λ^{SUTC} = Safe undetected by automatic diagnostics, but detected by test (CCF)

It is worth highlighting that the above discussion refers to the STR related to component internal failures (STR_i). Notice that the change in STR during test can have either a negative or positive effect, depending on the MooN architecture and the bypass used. This effect is independent from the adverse effect caused by the human error test-induced additional STR_{test} , which is discussed in the next section.

6.3.5. Test-induced STR

Eq. (5.27) established that the total STR of a component is the sum of the STR attributed to internal failure (STR_i), and the test induced (STR_{test}). The expression for estimation of STR_{test} was given in Eq. (5.28). The probability of test-induced spurious trip P_{r-trip} can certainly be influenced by the type of bypass used during test. A closed bypass in a 1ooN architecture could not trip the plant, but an open bypass in a MooN ($M>1$) actually could. This is not the only factor to consider indeed; otherwise P_{r-trip} would be equal to zero for a closed bypass. The human error influence on P_{r-trip} must include potential errors induced on the component being tested: e.g. *“it is possible to finish test and not exactly put electric cards to their places (sic) and start with test on another channel”*, but it can also contemplate errors that could affect other

components (not only the one being tested): e.g. during test of one channel “*another channel is taken off by human error for example*” (Cepin et al., 1994).

Section 5.8 demonstrated how the total STR of a parallel system could be modelled with fault trees (Fig. 5.10). There, the STR_{test} contribution is included into the fault tree as a basic event per component. In the example of Figure 5.10 the system is a two-component parallel arrangement, so the STR by test basic events are just summed up (i.e. OR gate).

In the case of spurious trips by internal failures in MooN voting systems the failure mechanism is more complex, not a simply OR gate anymore, and it changes according to the specific voting scheme, as it has been seen in the previous section. The question that arises here is whether the basic events related to STR_{test} should be subject to voting as well. As explained above, P_{r-trip} can include induced transients on both the component under test and possibly the other redundant components. It is therefore not clear enough whether the voting mechanism also changes the relation amongst the STR_{test} basic event in the fault tree. For example the question in a 2oo2 system would be whether the fault tree of the STR_{test} events would have an AND gate in a similar fashion as shown in Figure 6.12.

Several options arise. To quantify the total STR_{test} as a parallel system regardless of the MooN voting architecture could give a too large STR_{test} figure for MooN systems where $M > 1$. On the contrary, modelling STR_{test} using a fault tree with the specific MooN voting architecture may give a too small STR_{test} figure, because it could ignore that human error may affect both, the component under test or the components remaining in service. Since it is very difficult to determine at this stage if other components are affected and to what extent, a good strategy is to reach a compromise between the two options mentioned. An approach is using the voting scheme but based on the fault trees as they are reconfigured during test, during which time the test activity is truly prone to affect the STR. This situation is analyzed in Figure 6.15. The first column of the figure shows the fault trees corresponding to (independent) safe failure of the components. From this, the potential spurious trips caused during test (test-induced spurious trips) are derived, and shown in the second column.

For 1ooN architectures, the fault trees would remain without change due to the fact that the used bypass is closed, which is equivalent to a dangerous failure. Therefore, with relation to safe failures this basic event is equivalent to a value of zero ($A=0$ in Fig. 6.15) when modelling for STR_j . However, under this condition the operator still can erroneously open the bypass (instead of closing it) and provoke a spurious activation. Thus, even under test reconfiguration, all the components are considered.

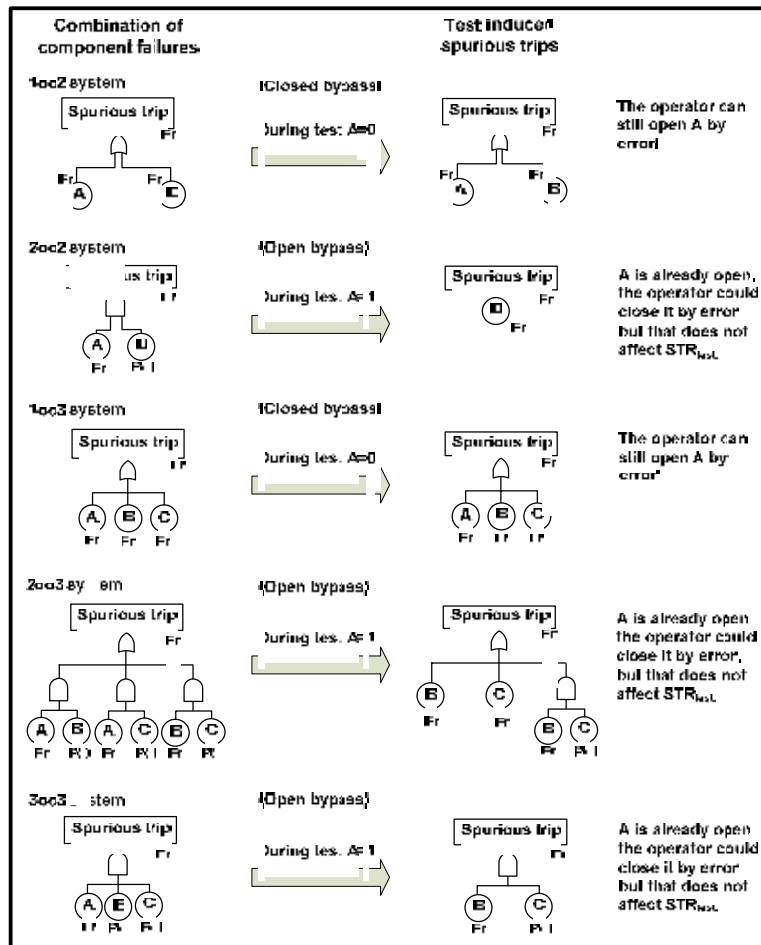


Figure 6.15. Derivation of test-induced STR (second column)

On the contrary, the bypass must be open for MooN architectures ($M > 1$), equivalent to a safe failure (thus $A=1$). This bypass could be left in place by error. However, since $A=1$ this is irrelevant to STR_{test} because this is equivalent to A being already spuriously failed. Therefore, the expression that would be used for quantify STR_{test} would be derived from the fault tree reconfigured by the test (compare Fig. 6.15 with 6.13-6.14). Notice that in this case STR_{test} would be larger than the one obtained using the fault tree during normal operation, but it is still smaller than simply adding the individual STR_{test} . This option is therefore, a middle ground solution between the two options initially mentioned. Table 6.9 derives the equations for quantifying the contribution of test-induced failures to the STR (based on Fig. 6.15). In the table λ^{STi} stands for test-induced spurious trips failure rate per component. Finally, table 6.10 presents a summary of formulae for quantification of STR.

Table 6.9. Contribution of test-induced failures STR_{test}

MooN	Contribution of test-induced failures
1oo2	$2\lambda^{STi}$
2oo2	λ^{STi}
1oo3	$3\lambda^{STi}$
2oo3	$2\lambda^{STi} + \lambda^{STi} \cdot P_{r-trip}$
3oo3	$\lambda^{STi} \cdot P_{r-trip}$
1oo4	$4\lambda^{STi}$
2oo4	$3\lambda^{STi} + 3\lambda^{STi} \cdot P_{r-trip}$
3oo4	$3\lambda^{STi} \cdot P_{r-trip} + \lambda^{STi} (P_{r-trip})^2$
4oo4	$\lambda^{STi} (P_{r-trip})^2$
1ooN	$N \cdot \lambda^{STi}$
2ooN	$\left[(N-1) + \left(\binom{N}{M} - (N-1) \right) P_{r-trip} \right] \cdot \lambda^{STi}$
MooN (2>M>N)	$\left[\binom{N-1}{M-1} (P_{r-trip})^{M-2} + \left(\binom{N}{M} - \binom{N-1}{M-1} \right) (P_{r-trip})^{M-1} \right] \cdot \lambda^{STi}$
NooN	$\lambda^{STi} \cdot (P_{r-trip})^{N-2}$

Table 6.10. Summary of formulae for quantification of STR

Architecture	State	Contribution	C_{MooN}
1ooN	Normal operation	$STR_{\lambda-IND}$	$N \cdot \lambda^{SN}$
		$STR_{\lambda-CCF}$	$Psf(CCF_{N-M+1})$
	Test	$STR_{\lambda-IND}$	$(N-1) \cdot \lambda^{SN}$
		$STR_{\lambda-CCF}$	$Psf(CCF_{N-M+1})$
		STR_{test}^*	$N \cdot \lambda^{STi}$
2ooN	Normal operation	$STR_{\lambda-IND}$	$\binom{N}{M} \cdot \lambda^{SN} \cdot Psf(SN)$
		$STR_{\lambda-CCF}$	$Psf(CCF_{N-M+1})$
	Test	$STR_{\lambda-IND}$	$\left[(N-1) + \left(\binom{N}{M} - (N-1) \right) Psf(SN) \right] \cdot \lambda^{SN}$
		$STR_{\lambda-CCF}$	$Psf(CCF_{N-M+1})$
		STR_{test}^*	$\left[(N-1) + \left(\binom{N}{M} - (N-1) \right) P_{r-trip} \right] \cdot \lambda^{STi}$
MooN (2<M<N)	Normal operation	$STR_{\lambda-IND}$	$\binom{N}{M} \lambda^{SN} \cdot Psf(SN)^{M-1}$
		$STR_{\lambda-CCF}$	$Psf(CCF_{N-M+1})$
	Test	$STR_{\lambda-IND}$	$\left[\binom{N-1}{M-1} Psf(SN)^{M-2} + \left(\binom{N}{M} - \binom{N-1}{M-1} \right) Psf(SN)^{M-1} \right] \cdot \lambda^{SN}$
		$STR_{\lambda-CCF}$	$Psf(CCF_{N-M+1})$
		STR_{test}^*	$\left[\binom{N-1}{M-1} (P_{r-trip})^{M-2} + \left(\binom{N}{M} - \binom{N-1}{M-1} \right) (P_{r-trip})^{M-1} \right] \cdot \lambda^{STi}$
NooN	Normal operation	$STR_{\lambda-IND}$	$\lambda^{SN} \cdot Psf(SN)^{N-1}$
		$STR_{\lambda-CCF}$	$Psf(CCF_N)$
	Test	$STR_{\lambda-IND}$	$\lambda^{SN} \cdot Psf(SN)^{N-2}$
		$STR_{\lambda-CCF}$	$Psf(CCF_{N-1})$
		STR_{test}^*	$\lambda^{STi} \cdot (P_{r-trip})^{N-2}$

*Applies only during test

6.3.6. Application example

The STR model is tested in a group of MooN redundant subsystems (where $N \leq 5$) composed of conventional transmitters. The device specification corresponds to the electronic PT in Table 5.1. Simultaneous test is assumed, with a mission time of 10 years, and a proof test coverage over safe failures $\epsilon^{ts} = 50\%$.

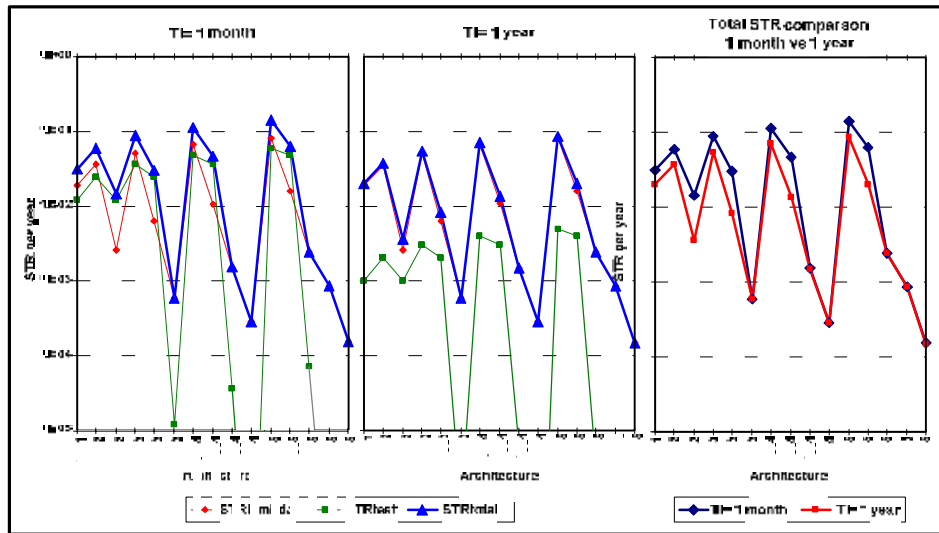


Figure 6.16. Evaluation of STR for several MooN architectures

Results are shown in Figure 6.16. The first two figures show the STR_{λ_s} , STR_{test} and total STR for one-month and one-year test interval respectively. The third graph presents the comparison of total STR for the two cases. Observe the first plot where $TI=1$ month. STR_{test} is larger than STR_{λ_s} for some architectures, which in turn increments considerably the total STR. This demonstrates the importance of including STR_{test} in the quantification. Observe in the second plot, however, that with a larger TI (1 year), STR_{test} is usually smaller than STR_{λ_s} , and the total STR is practically equivalent to STR_{λ_s} , which indicates that with a larger TI the negative effects of the test have less importance. The third plot compares the total STR for both cases. STR for $TI=1$ month is consistently higher or equal to $TI=1$ year, which confirms the negative effects of too frequent testing. However, observe that for architectures where $M \geq 3$ the effect of test on the STR becomes negligible, and the STR for both cases become practically equal.

6.4. APPLICATION TO OPTIMIZATION OF SYSTEM DESIGN

6.4.1. Description of the problem

Both models developed above, for PFD_{avg} and STR, are applied in this and the next section to optimization of SIS. In this section, the application case is for optimization of system design. The application case is the protective function against high pressure and temperature of a chemical reactor. Figure 6.17 shows the system. It is assumed that in every case the voters are perfect. It is also assumed that the PT and TT must be both fully functional for the safety system to be working. Therefore, the four subsystems are to be modelled in a logical series structure. The data of the problem is shown in Table 6.11. The case study seeks to explore whether introducing voting architectures as an alternative to simple parallel redundancies in the optimization of design would enable the achievement of better trade-offs between the two dependability attributes (PFD_{avg} and STR), and therefore to attain a better LCC. This can be achieved optimizing the N and M indices of the MooN voting system. Two optimization cases are run in order to contrast their results.

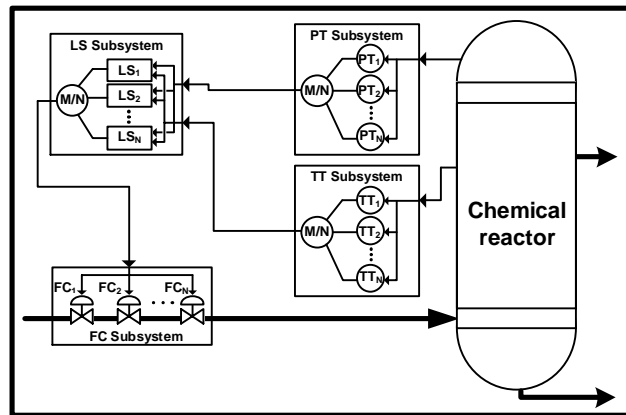


Figure 6.17. Chemical reactor protection system with MooN subsystems

- Case 1. Optimization of system design with option of MooN architectures. All subsystems, excepting the FC, can be subject to voting. The decision variables are: the type of components for the subsystem (three choices per subsystem, distinguished and type 1, 2, 3, excepting for FC that are two), the number of components N (up to 5), and the number of M voting components ($M \leq N$) (for PT, TT and LS). The search space has approximately 4.22×10^6 potential solutions. The vector of decision variables is therefore:

$$\mathbf{x} = \{Type_{PT}, N_{PT}, M_{PT}, Type_{TT}, N_{TT}, M_{TT}, Type_{LS}, N_{LS}, M_{LS}, Type_{FC}, N_{FC}\} \quad (6.37)$$
- Case 2. Optimization of system design with only parallel architectures. The same as above, but all subsystems are constrained to $M=1$.

Table 6.11. Dependability and Lifecycle Cost data

Subsystem Type	PT			TT			LS			FC		
	1	2	3	1	2	3	1	2	1	2	3	
	Smart transmitter	Conventional Electronic transmitter	Switch	Smart transmitter	Conventional Electronic transmitter	Switch	Safety PLC	Standard PLC	Air operated ^b	Hydraulic operated ^b	Motor operated ^b	
λ^{SD} (%)	69.2/31.8	56.0/51.1	10/10	97.9/7.5	86.9/45.5	10/10	100/81.25	45/60	0/25	0/20	0/10	
λ^{SD} (x10 ⁻⁶ /hr)	0.265	1.21	0.68	5.05	6.5	0.92	3.46	1.77	0	0	0	
λ^{SU} (x10 ⁻⁶ /hr)	0.118 ^a	0.95	6.13	0.11 ^a	0.98	8.30	0	2.17	3.94	3.17	9.17	
λ^{SD} (x10 ⁻⁶ /hr)	0.048	0.97	0.41	0.026	1.57	0.76	0.026	2.89	0.84	1.09	0.79	
λ^{SU} (x10 ⁻⁶ /hr)	0.103	0.93	3.70	0.322	1.88	6.84	0.006	1.92	2.51	4.35	7.11	
λ^T (x10 ⁻⁶ /hr)	0.534	4.06	10.92	5.508	10.93	16.82	3.492	8.75	7.29	8.61	17.07	
Type	B	A	A	B	A	B	B	B	A	A	A	
SFF (%)	80.7	77.09	66.12	94.15	82.79	59.33	99.83	78.06	65.57	49.48	58.35	
C _{purchase} (\$)	4844	2306	500	2560	1406	500	3000 ^c	2500 ^c	6940	6400	6200	
Lifecycle cost data:						Other data:						
Design/install/commissioning PLC= 10,320 (\$)						Cost of rack of Safety PLC: 31000 (\$)						
Maintenance PLC= 960 (\$/event)						Cost of rack of Standard PLC: 20500 (\$)						
Test PLC= 240 (\$/event)						Repair time=8 (hrs)						
Design overall instrumentation= 3,060 (\$)						Shut down time= 24 (hrs)						
Installation/commissioning per instrument=600 (\$)						Test Interval= 1 (year)						
Maintenance per instrument= 240 (\$/event)						Plant risk without SIS=8.55 (x10 ⁻³ /yr)						
Test per instrument= 60 (\$/event)						$\beta=10\%$ ^e						
Repair cost per instrument & PLC= 60 (\$/hour)						Notes:						
Spares per repair= 25% component cost						^a This failure mode does not cause spurious trip, only quantified for SFF						
Loss of production= 2,000 (\$/hour)						^b Includes actuator						
Start up cost= 1800 (\$)						^c The PLC must be bought with a rack which cost is indicated above						
Catastrophic loss=150x10 ⁶ (\$)						^e Apply for all cases, except for β^{SD} of: PTA=5%, TTA=2%, TTB=5%						
SIS life= 15 (years)												
Discount rate=0.05												

6.4.2. Implementation of the solution

Modelling of the dependability objectives is made with the models developed in Sections 6.2 and 6.3. The fault tree for PFD(t) at subsystem level is constructed based on the guidance given by Table 6.5. The fault tree at system level includes house events that select amongst different architectures for each subsystem, as it is shown in Figure 6.17. The tree only exemplifies the PT subsystem, while the other subsystems are developed in the same fashion.

The quantification of PFD(t) at subsystem level is based on the model given by Eqs. (6.10-6.19). The failure rates for the basic events are determined by Eqs. (6.20-6.25). The modification factors C_{MooN} for each architecture can be determined based on Table 6.6, and their actual values taken from Table 6.2. Same as in Chapter 5, the PFD(t) is evaluated every hour for the entire system operating life, and then these point values are averaged (Eq. (5.20)).

The fault tree for STR follows the philosophy of Figure 6.18 regarding the application of house events, and using of course the adequate selection of logic gates. At subsystem level the fault trees can be constructed and quantified following Table 6.10 (the values for modification factors are taken from Table 6.62). Remember that, according to Eq. (5.27), the total Spurious Trip Rate of each subsystem is composed by the sum of spurious trip caused by internal failures and test-induced ($STR_T=STR_\lambda+STR_{test}$). The fault tree follows the general structure seen in Figure 5.10. These contributions are reconfigured during testing, determined by Table 6.10, and summed as indicated by Eq. (6.34). Failure rates of basic events can be determined applying

Eqs. (6.28-6.30) and (6.37-6.38), and the test-induced share by Eq. (5.28).

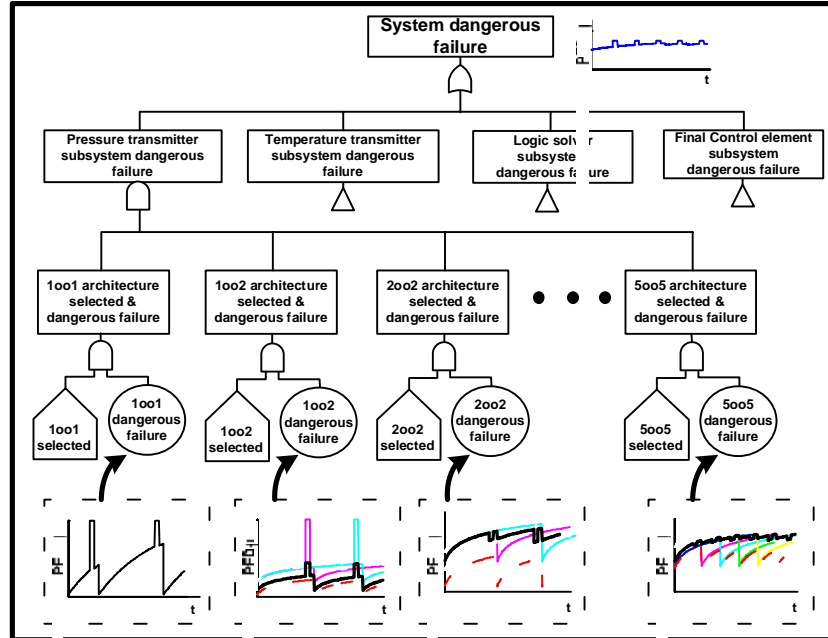


Figure 6.18. Fault tree for quantification of PFD with optional MooN architectures

Modelling of LCC is made using the model presented in Chapter 4 (Eqs. (4.12-4.22)) quantifying the cost of repair time with Eq. (5.35).

The implementation of the optimization algorithm is the same as in Chapter 5. The NSGA-II algorithm is used with controlled elitism, following the structure showed in Figures 5.13-5.14. The codification is made with integer numbers and the parameters are used as indicated in Section 5.10.3.

6.4.3. Discussion of results

The Pareto-optimal sets obtained from both cases are compared in Figure 6.19. The plots PFD_{avg} vs STR and PFD_{avg} vs LCC clearly show that the solutions of both fronts with the same level of PFD_{avg} have lower LCC and lower STR in the optimal set of Case 1, where the option to choose MooN architectures was available. For the graph STR vs LCC this advantage is not that clear. However, if one solution of Case 1 is picked, it can be seen that for the same LCC a solution in Case 2 would have a higher STR. In order to make a more rigorous verification of the advantage of Case 1 over Case 2, the solutions belonging to both sets were put together and re-ranked to find only the non-dominated individuals. From all the set (202 solutions altogether), Case 2

provided only three optimal solutions (at the extreme end of the front, where $PFD_{avg} \sim 1 \times 10^{-4}$), and one of them actually belonged to both sets (Cases 1 and 2). The other two are still a sub-set of Case 2. Thus, they should still be locatable in the front in further runs of this optimization case. In general, the optimal set with MooN architectures dominated the only-parallel one.

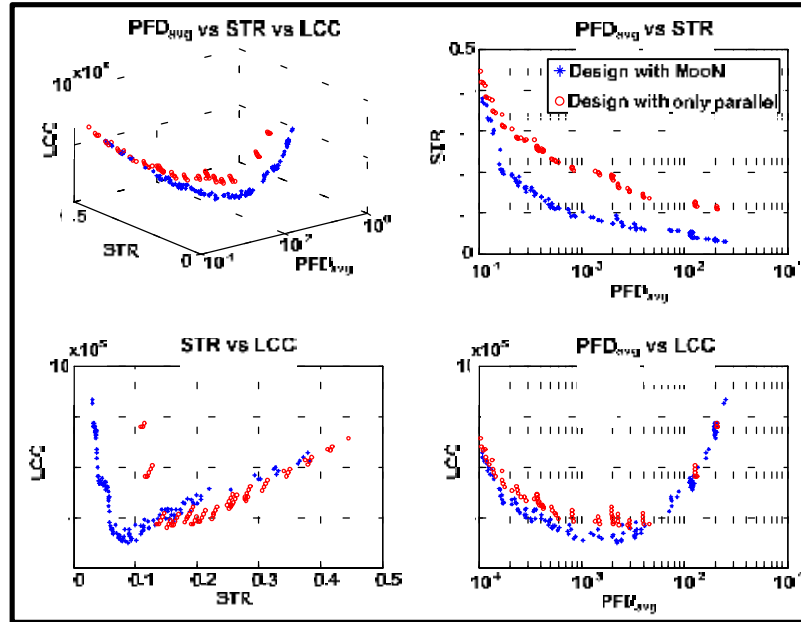


Figure 6.19. Comparison of optimal sets of both optimization of design cases

Another important observation from Figure 6.19 is that in these cases the PFD_{avg} is not always in conflict with LCC in the Pareto-optimal front. There is a region where PFD_{avg} and LCC hold a harmonious relationship ($PFD_{avg} < 2.2 \times 10^{-3}$), and another where they are in conflict ($PFD_{avg} > 2.2 \times 10^{-3}$). This is a result of the balance of costs achieved with the changing design: procurement vs operation costs and safe vs dangerous risk costs. On the other hand, STR is also in a dual relationship with LCC in the front, which change around $STR \sim 0.087$. Finally, it can be seen that PFD_{avg} and STR are consistently in conflict. Another feature to observe is that a region of high saturation (PFD_{avg} vs LCC) on the left side of the graph is not present as in the cases of previous chapters. A high saturation point appears only if the PFD_{avg} scale is changed to linear.

Observe now Figure 6.20, in which the full results are shown for both cases. In this figure, the particular combination of decision variables and resulting objectives are shown for each solution of the optimal set. It is notable that the optimizer consistently selects the components with better specifications despite their higher acquisition cost (mostly Type 1, see Table 6.10). These components have overall lower failure rates and mostly higher diagnostic coverage. This means that they have a reduced impact in both PFD_{avg} and STR (i.e. lower values) in

comparison with the other types of components available. This in turn impacts positively on the overall LCC. Therefore, their benefits in terms of reduced operational costs and risk costs, by both loss of production and loss of safety (i.e. catastrophic hazards) outweigh the expenditure caused by their high initial acquisition cost.

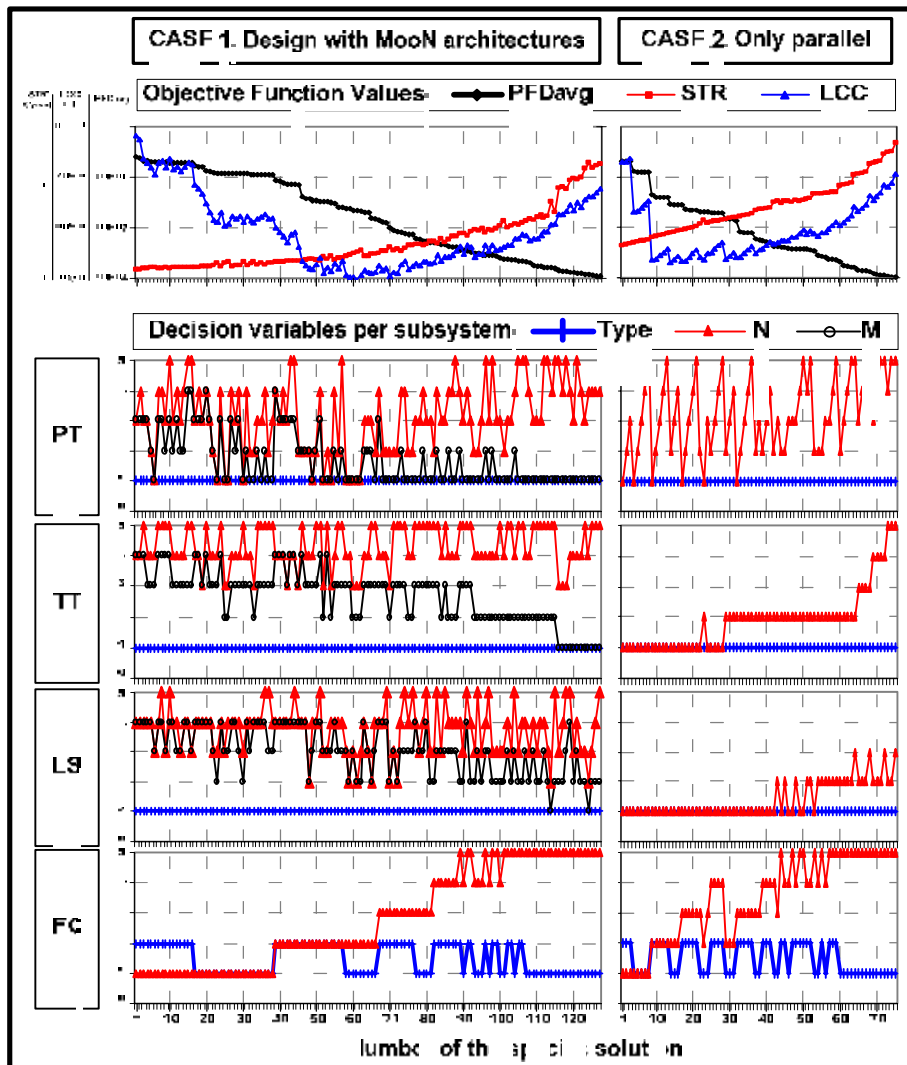


Figure 6.20. Results obtained from both design optimization cases

Comparing the solutions from both cases, as shown in Figure 6.20, it is clear that optimization with MooN offers many more options to the optimizer. In the only-parallel case, improvements in $PFDAvg$ are gained by adjusting the FC subsystem, then the PT. In these two subsystems lie most of the trade-offs. Once no more improvements are possible changes in the TT and LS

subsystems are explored. It can be seen in the optimization with MooN that the trends of the variables of the four objectives are much less flat than the only-parallel case, meaning they change more dynamically to achieve optimal trade-offs.

In Case 1, the FC subsystem (which has only parallel redundancy) dominates the changes in PFD_{avg} and LCC. This is observable in Figure 6.21, where the solutions have been ordered ascending number of FC's components. Notice the step-like changes in those variables when the number of components, or the type, are changed. This suggests that the improvements in PFD_{avg} are largely acquired by increasing the parallel redundancy level of the FCs. This, however, also increases the STR.

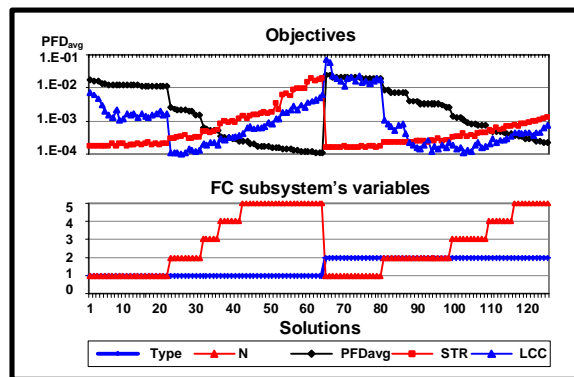


Figure 6.21. Case 1. Objectives in relation to the FC subsystem

It is important to notice how, from left to right, decreasing the PFD_{avg} requires an increment in the number of components in the subsystem N and a reduction in the M voting components. This is easy to spot in the variables of the PT and TT subsystems. These changes, however, affect the STR in the opposite sense. Therefore, the option of being able to choose between different levels of M voting components provides a better balance of PFD_{avg} against STR. Observe both ends of the PT and TT subsystems. At the far left, with the lowest STR, the voting is NooN. On the contrary, at the far right, with the lowest PFD_{avg} , the voting is 1ooN.

Clearly this trading between the variables N and M also allows balancing of the total LCC associated with the system. Observe that the top graph showing the objectives reiterates the trend in which the PFD_{avg} and STR have a dual relationship with the LCC in the different halves of the graph, the transition of which is marked by solution 61 which has the lowest LCC. The likely explanation is that the risk cost associated to the STR (C_{STR} in Eq. 4.23) becomes the dominant factor of the LCC after solution 61. Observe the set of solutions 1-61. Trade-offs are both in N and M values of the three MooN subsystems (PT, TT and LS). Also the number of

solutions with $M > 1$ (i.e. non-parallel subsystems) is very high, being also with M quite close to N in most of the cases. It was seen in previous sections that, with a fixed number of components N , increments in the voting components M increases the PFD_{avg} (Fig. 6.10) but reduces the STR (Fig. 6.16). It is thus clear that a low N/M ratio keeps the STR low (>0.087), which results in a reduction of PFD_{avg} and also reduces the LCC. On the contrary, for solutions 62-127 the number of solutions with $M > 1$ is largely reduced. The trade-offs increasingly depend on the number of components N . Observe that, in general, the values of M are smaller than for solutions 1-61. What is more, the PT subsystem has mostly $M=1$ (in solutions 62-127), and the number of components in the FC subsystem is increased significantly. This means that many more solutions in that group (65-127) have subsystems which are simple parallel and with a higher number of components. Also the number of subsystem with $M > 1$ have a larger N/M ratio. This situation increments the STR consistently above 0.087, becoming a dominant factor of the LCC. Observe that the LCC practically follows the trend of the STR for solutions 62-127. In conclusion: systems with a higher level of voting (lower N/M ratio) achieve lower values of STR greatly benefiting the LCC. However they have higher values of PFD_{avg} . On the contrary, increasing the number of components and using more parallel architectures decreases PFD_{avg} (i.e. safer systems) but increases the STR and, to some extent, the LCC.

Finally, Figure 6.22 presents the values of Safety Integrity Levels (SIL) achieved by the solutions. The SIL corresponding to the system PFD_{avg} and their architectural constraints are shown. Remember that the final SIL is the one with the lowest values from these two. Observe that in both optimization cases, for systems achieving SIL 2 or 3 by PFD_{avg} , the architectural constraints limit the SIL claimable to one inferior level many times. The opposite happens for systems with SIL 1 by PFD_{avg} . This is surely a constraint for the decision maker to consider.

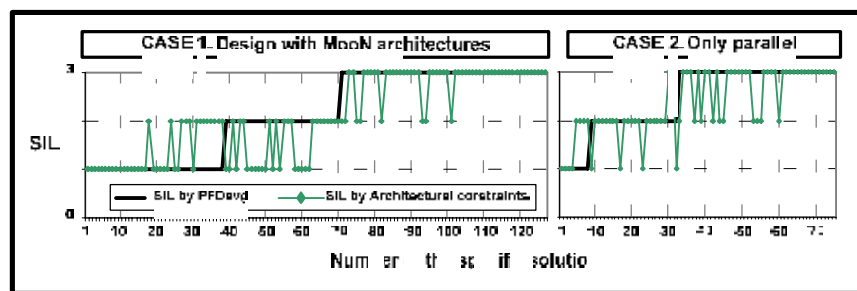


Figure 6.22. SIL achieved by the solutions of the optimal sets of Cases 1 and 2

6.5. APPLICATION TO OPTIMIZATION OF TESTING POLICIES

6.5.1. Description of the problem

This second case study is about application of the models to optimization of testing policies. The design of the reactor protective system has been defined as shown in Figure 6.23. The technology used in the subsystem is as follows: conventional pressure transmitter, smart temperature transmitter, safety PLC and air operated valves. All data has already been provided in Table 6.10.

In Chapter 5 it was verified that proof testing has both positive and negative impact on the system dependability and LCC. This case study aims to explore whether the optimization of testing policies where additionally the voting scheme (M) of some MooN subsystems can be chosen could permit the implementation of better testing policies, counteracting the negative impact of testing while having the same benefits in terms of safety and cost. This requires optimization of the testing policy at subsystem level together with the number of M voting components, subject to the constraint $M \leq N$.

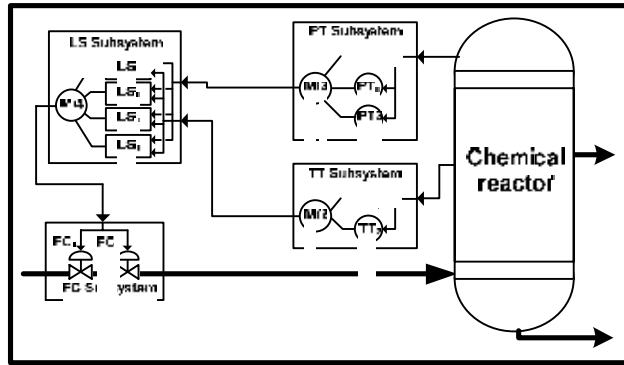


Figure 6.23. Safety system for the testing optimization case

Two optimization cases are carried out:

- Case 3. Optimization of testing policies with option of MooN architectures. The testing policies of the subsystem are optimized in the same fashion as in the previous chapter (see Section 5.10). The TI is bounded between 30 and 365 days (730 for the LS), and T_{PIS} between 0 and TI. The multiplication factor P is a number between 0 and 1000 (then multiplied by 1×10^{-3} so that $P < 1$). In addition the number of voting components M of the PT, TT and LS subsystem can be chosen (subject to $M \leq N$). The total number of potential solutions is 1.68×10^{31} . The decision variables vector is then:

$$\mathbf{x} = \{M_{PT}, TI_{PT}, Tp_{1PT}, P_{PT}, M_{TT}, TI_{TT}, Tp_{1TT}, P_{TT}, M_{LS}, TI_{LS}, Tp_{1LS}, P_{LS}, TI_{FC}, Tp_{1FC}, P_{FC}\} \quad (6.38)$$

- Case 4. Optimization of testing policies only. The same as above, but all subsystems have just simple parallel redundancy.

6.5.2. Implementation of the solution

The theory of testing policies and their optimization has been detailed in Chapter 5, and the reader needs to be familiar with its content to be able to understand the implementation of this case study. The execution of the case study is also made in a similar fashion as for the one in the previous section. For Case 3, house events in the fault trees enable the change in voting components M when selecting amongst different architectures constrained by the number N of each subsystem. This is illustrated in Figure 6.24 for the PFD_{avg} . Construction of the fault tree for STR follows the same philosophy. Variations in the testing variables (TI, Tp1 and P) are incorporating by feeding them into the equations for dependability modelling.

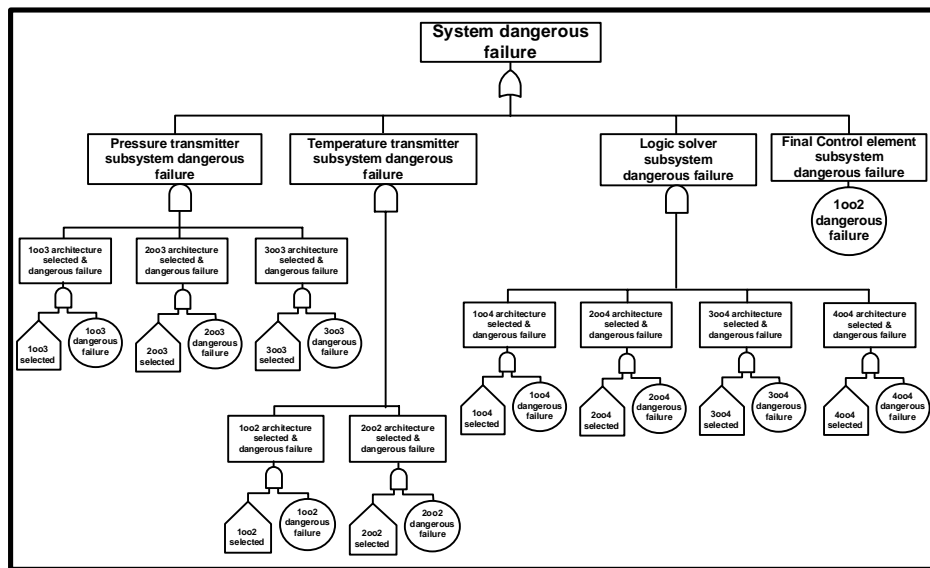


Figure 6.24. Fault tree for PFD_{avg} of Case 3

6.5.3. Discussion of results

The Pareto-optimal sets obtained from both optimization cases are compared in Figure 6.25. The first notable feature to observe is the domination of the optimal set of Case 3 over the optimal set of Case 4. It is clear from observing the graphs PFD_{avg} vs STR and PFD_{avg} vs LCC that, for the same levels of PFD_{avg} , the costs in terms of STR and LCC are lower for Case 3 (optimization of testing together with voting M). It is also shown, in the STR vs LCC graph, that solutions with the same level of LCC have lower STR in Case 3. An exercise in which the solutions of both optimal sets were put together and re-ranked to obtain the only optimal ones

showed that, from a set of 149 solutions, only three of them belonged to the optimal set of Case 4. Since these belong also to the search space of Case 3, it is believed that new runs of the algorithm could find them. Therefore, it can be concluded that the optimization of testing polices together with the level of voting M gives results that dominate those from optimization without M .

Analyzing the relationships between the objectives in the optimal set of Case 3, it is shown that the PFD_{avg} has a dual relationship with LCC. There is a region of harmonious relationship ($PFD_{avg} > 1.09 \times 10^{-3}$), and another region of conflicting relationship ($PFD_{avg} < 1.09 \times 10^{-3}$). The PFD_{avg} and STR (as with the cases analyzed in previous chapters) have a consistently conflictive relationship. The STR and LCC tend also towards conflict.

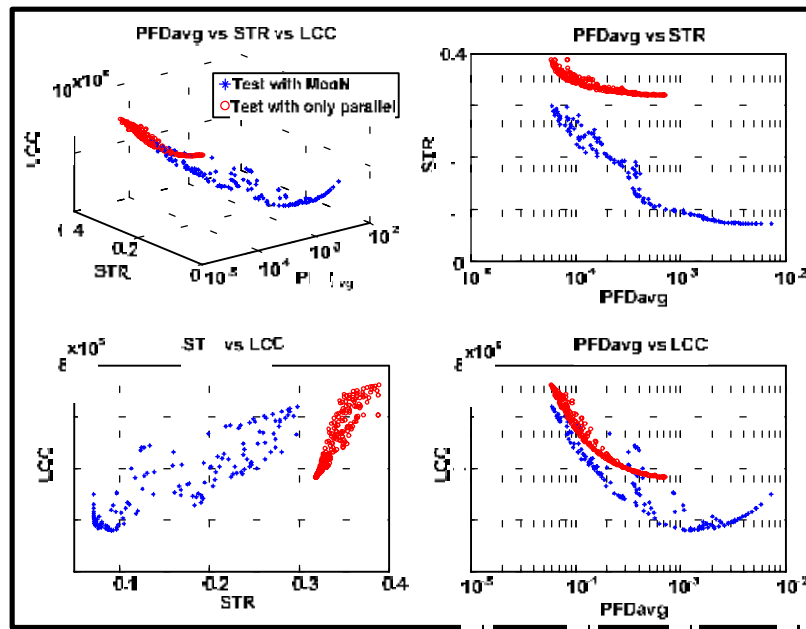


Figure 6.25. Comparison of optimal sets of both optimization of testing cases

Figure 6.26 shows the complete results of both optimization cases, including the values of the objectives and decision variables of each solution belonging to the Pareto-optimal sets. The two graphs are put side by side to allow the reader to easily compare the two optimization cases. With this one can easily identify the trends and trade-offs amongst the variables and subsystems. Observe firstly the results given by Case 4. The PFD_{avg} is reduced by primarily lowering the TI of the FC subsystem (the time to first test T_{p1} is already low). Notice that the trend is to reduce the TI at the lowest possible level (30 days). Once this is done, the TI of the TP and TT subsystems are also lowered to achieve further reduction in the PFD_{avg} . The

relationship of the PFD_{avg} versus the LCC and STR is always conflictive in this optimization case. Therefore, the action described consistently raises these two objectives. Recall that the FC subsystem is the one with the lowest level of redundancy (1oo2) and with the less reliable components. Therefore the reduction of PFD_{avg} depends largely on its testing policy. On the contrary, the LS that has high redundancy (1oo4) and highly-reliable components remains with a high TI; i.e. above 400 days (low testing frequency). Also notice that the multiplication factor P (explained in Eq. 5.25) of every subsystem, excepting LS, tends towards a high value, i.e. towards full uniform staggering.

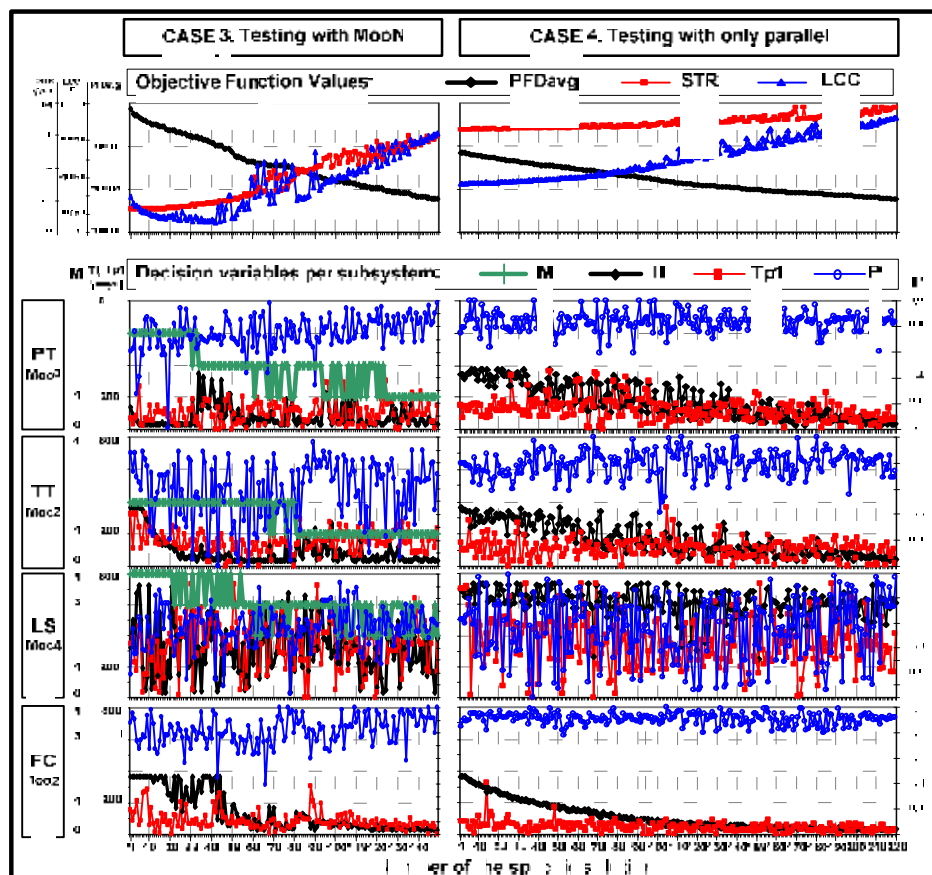


Figure 6.26. Results obtained from both testing optimization cases.

Now observe the results from Case 3. In comparison with Case 4, solutions with the same value of PFD_{avg} have lower LCC and STR values. This is notably evident for the STR, where the difference of values can be noticed at a single glance at the upper part of Figure 6.26. Note, however, that the TI and the T_{pl} values of the PT, TT and FC subsystems remain at very low levels. Their values are even lower than for the system with only-parallel redundancies (Case 4).

This means higher testing frequencies and earlier in the system lifecycle. Nevertheless, in an opposite sense, the multiplication factor P (in Case 3) has lower and more changing values than in Case 4. This means that the distribution of the test events along the test interval has a lower trend to be uniform (full staggering).

Remember that the only difference between cases 3 and 4 lies in the fact that Case 3 permits voting (i.e. M is a decision variable) and Case 4 is limited to parallel redundancies (i.e. $M=1$). Therefore, it is possible to that say introducing voting, permitting M to be varied between 1 and N , causes of lower values of P in the optimal set of Case 3. This in turn suggests that trade-offs are made between the level of voting M and the level of test staggering (determined by P) to achieve lower values of PFD_{avg} . This strongly suggest that by trading between the voting level M and the test staggering P (with low TIs), low levels of PFD_{avg} can be achieved with lower values of STR and LCC rather than when varying only the testing parameters (with $M=1$).

This conclusion motivates a deeper analysis. It is clear that introducing voting in the subsystem (i.e. $M>1$) lowers the level of staggering in the optimal set (the variable P). Even in the FC subsystem that has only parallel redundancies the values of P are also lower (than in Case 4). Therefore, the introduction of voting in some subsystems causes lower levels of P in all subsystems. Also notice that even for solutions 124 to 149 (Case 3), where $M=1$ for all subsystems except for LS, the values of P are still lower than in Case 4. Also observe that in Case 3 the same values of PFD_{avg} in both cases correspond to lower STR and LCC figures (than in Case 4). All these remarks lead to the conclusion that the trade-offs between M and P are at the system level. It is however difficult to establish what is the specific trade-off at subsystem level, since there is no a recognizable clear exchange between M and P on a case-by-case basis.

Observe also that the level of reduction of P is more prominent in the TT and FC subsystems, in that order. This can be explained observing Table 6.11, where it is seen that the total failure rate per component λ^T is higher in the smart transmitter and the air operated valves than in the conventional pressure transmitter and the safety PLC (the technologies used in this case study). In addition, compare the safe and dangerous failure rates of the smart temperature transmitter against the conventional pressure transmitter. The PT has higher dangerous failure λ^D rates than the TT, but the TT has significantly higher safe failure rates λ^S than the PT. Thus, it can be said that the staggering P is lowered in the TT to decrease the STR (which is affected by higher λ^S).

Observe now in the results given by Case 3 that the test intervals are also lower than in Case 4; i.e. higher testing frequency. This is true for all subsystems, although more notable in the field instruments (PT, TT and FC). This observation can be reinforced by comparing the group of

solution that are between the same band of values of PFD_{avg} in both cases. These are solutions 50-149 in Case 3 against the complete set of Case 4. The TIs in Case 3 solutions are consistently lower. Therefore, it is evident that there is also a trade-off between TI and P. Since proof testing is aimed at dangerous failures, this suggests that once voting is introduced in the form of the variable M, the optimizer chooses to test more frequently the dangerous failures in all subsystems (with lower TIs) in exchange of less staggered testing especially in the subsystems with higher safe failure rates.

In addition, also trade-offs between the variables M and TI can be seen in some specific solutions of Case 3. For instance, observe solutions 94-96 in the PT subsystem. The value of M decreases and TI increases simultaneously. Also in solutions 125-149 of the PT subsystem the variable M becomes equal to 1 at the same time that the values of TI are raised. For the TT subsystem there is a rise in TI values after solution 81, where M becomes equal to 1. Thus, it is clear that reduction of M cause TI to rise.

Considering the analysis of the previous paragraphs, it can be said that introducing voting levels as the variable M in the subsystems enables trade-offs between this and the TI and P variables. These are generally in the form of higher testing frequency in exchange of less staggered test. This trading in turn allows achievement of similar values of PFD_{avg} with lower STR and LCC than with only parallel architectures.

It is customarily believed that full staggering (the uniform distribution of the test events of the N components along the test interval; i.e. TI/N) is the best option. It can be seen in this exercise, however, that reducing the level of staggering can actually bring benefits when combined with the changes in the voting scheme M and the test intervals. Nevertheless, it is important to notice that very low staggering (i.e. low values of P) can affect negatively the LCC, which can be seen in solutions 71, 75 and 90 that present high sudden peaks in LCC. As it was seen above, in Case 4 the reduction in PFD_{avg} is primarily achieved by lowering the TI and T_{p1} of the FC, TP and TT and FC subsystems. In Case 3, this is achieved by reducing in addition the number of voting components M towards simple parallel ($M=1$) of the TP and TT subsystems. It is evident that the addition of the voting level M as a variable in the optimization of testing policies opens the possibility of new interesting trade-offs that enable the achievement of overall better solutions.

The conclusions drawn above are believed to be applicable to low-demand safety systems under similar conditions as those assumed in this thesis; especially constant and low failure rates. It is necessary to note that the results are also dependent on the values given for the probability of test induced spurious trip per component (P_{-trip} in Eq. 5.28). Based on the evidence provided by

this single case study it is difficult to determine whether this is a general design lesson for safety systems. It is however a promising result that could be validated as a generalization through further research.

Consider now the trend obtained for the LCC in Case 3. The graph of objective's values for shows that the trend of the LCC is to be firstly reduced and later incremented (congruent with the dual relationship PFD_{avg} vs LCC described above). This may be explained by the changing influence of the costs caused by the STR over the LCC. It is suggested that by varying M together with the testing policy variables permits lowering of the STR to such a level that it does not considerably affect the LCC (solutions 1-42). It is in this region that the LCC is in harmony with the PFD_{avg} . Subsequently, the STR increases, when reducing the values of the M variables, in such a way that seems to exert considerable influence over the LCC, and even dominates it when all $M=1$ (i.e. all subsystems have parallel redundancy). Thus, the PFD_{avg} and LCC become conflicting objectives (solutions 43-149). It is difficult to determine the exact influence of the variables over the LCC given that they are numerous. However, a useful analysis can be made by reordering the objective's value graph, as made in Figure 6.27.

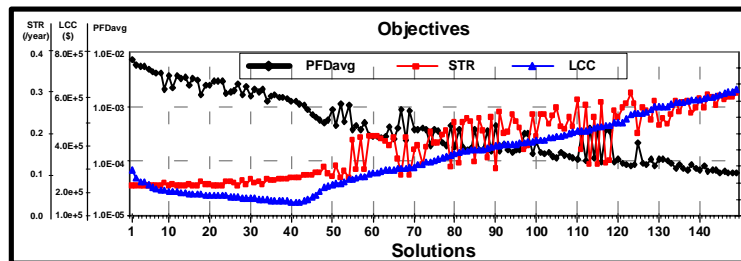


Figure 6.27. Optimal set objective's values of Case 3 reordered by LCC

In Figure 6.27 the optimal set is ordered in descending LCC for the group of solutions 1-42 (because solution 42 is the one with lower LCC and where its trend changes direction), and ascending order for solutions 43-149. It is important to clarify that solutions in Figure 6.27 correspond to the ones in Figure 6.26 only as groups 1-42 and 43-149, not necessarily individually. For the first group 1-42 (Fig. 6.27) changes in PFD_{avg} and STR along the LCC are smooth. These changes correspond to the group of solutions 1-42 in Figure 6.26. Practically the M values of the PT and TT subsystems are high and constant. For the second group 43-149 (Fig. 27), where the LCC raises steadily, the trend of the PFD_{avg} is to descend and the trend of the STR is to ascend. There are regions of high trade-offs between these two variables that permit the LCC to keep rising steadily. These trade-offs are notable for example in the group comprised by solutions 110-120 in Figure 6.27. Clearly this trading between the costs related to both PFD_{avg} and STR is important to keep the LCC trend stable. In Figure 6.26 this can be seen

for example in the region of solutions 60-80 and 95-125, where M is changing together with P , and the changing rate of STR and LCC is non-smooth.

The group of solutions 1-42 (Fig. 6.26) where changes in the objectives are pretty smooth correspond to the voting level M of PT and TT being held constant. This region is not explored in the optimization with only parallel architectures. Observe that the highest PFD_{avg} value achieved in Case 4 is 7.13×10^{-4} . In Case 3 solutions 1-50 have higher values than this. Thus exploration in the area of SIL 2 ($PFD_{avg} > 1 \times 10^{-3}$) is not present in Case 4. This is another interesting feature of the optimization adding M as variable.

Finally, Figure 6.28 presents the SIL levels achieved in both optimization cases. The SIL 3 level achieved by architectural constraints do no change in Case 4 because the only-parallel architecture of the system does not change. However, after solution 146 the PFD_{avg} corresponds to a SIL 4 level. On the contrary, the SIL by architectural constraints in Case 3 changes between levels 2 and 3. It cannot go higher than the level 3 (achieved with only parallel redundancies) because increasing M lowers the PFD_{avg} . The problem of diverging SIL levels (achieved by PFD_{avg} and architectural constraints) is again present as in previous optimization cases. This needs to be considered when choosing a specific architecture.

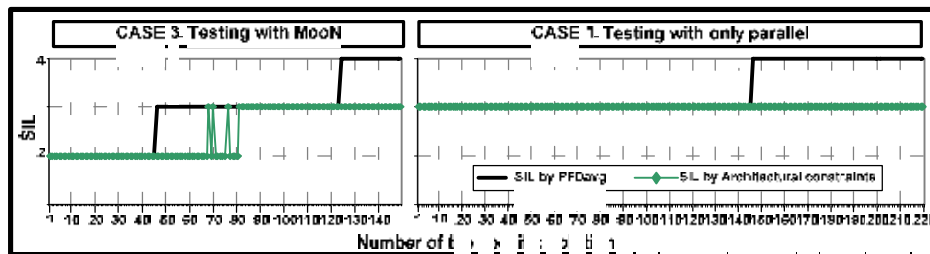


Figure 6.28. SIL achieved by the solutions of the optimal sets of Cases 3 and 4

6.6. CONCLUDING REMARKS

This final chapter puts together the work done for modelling and optimization of SIS (including the required level of modelling detail) of previous chapters and extends it for inclusion of MooN voting architectures into the redundant subsystems.

An extensive part of the chapter has been dedicated to the further development of the $PFD(t)$ and the STR models in order to enable them for quantification of MooN systems' dependability. This has required a large amount of work on the analysis of the system reconfiguration when changing from the normal operation state to testing of one component dictated by the fact that

the component being tested (with the system on-line) is taken out of service. This is determined by the operational philosophy, which establishes the type of bypass to be used depending on the specific MooN voting architecture. Therefore, given the particular bypassing philosophy, the combinations of independent failures, and specially the CCF contribution, change; this is reflected in the fault tree of the system.

The PFD(t) model for parallel architectures (developed in Chapter 5) did not change significantly for MooN systems. The major changes to accommodate MooN architectures reside in how the independent and common cause failure events are quantified and how their combinations are structured in fault trees. The STR model also considers the adverse effects due to the proof testing activity, and how this contribution changes during the test itself. Thus, the analysis completed here has provided a set of formulae that guide the construction of fault trees and determine the quantification of failure rates. Both models are based on approximations by quantification of first-orders cut sets, which in turn are based on quantification of CCF by using the β factor model.

The developed models have been applied to two optimization cases. One for optimization of system design (redundancy allocation and component selection) with the option of choosing the number of M voting components. The second case is about optimization of testing policies with also the addition of the voting level M as a decision variable. The case studies have been solved following the methodology of previous chapters based on fault trees with house events and multi-objective genetic algorithms. It was seen that the set of solutions given by the optimization cases with the variable M dominated the set of solutions of the optimization cases without it. In other words, for the same values of PFD_{avg} , lower levels of STR and LCC were achievable. These results demonstrated that the inclusion of the voting level M as an additional decision variable (in both optimization cases) permits the achievement of better trade-offs than when this is not considered. In addition, this has allowed extension of the search space considerably. For the test optimization case it even pushed the results to a region not explored by the optimization with only parallel architectures (i.e. SIL 2). In the optimization of system design it opens up the possibility of new trade-offs between the number of components N and the voting components M. In general, to increase the number of components M (with a fixed total number N) raises the PFD_{avg} but reduces the STR. This trading permits the achievement of designs with a better balance between these two dependability measures. In addition, this trade-off has a significant, potentially positive, impact on the system LCC.

Important trade-offs have also been found in the optimization of testing policies between the voting level M, the test intervals and the level of staggering of the test events (i.e. distribution of

the test events along the test interval, determined by a multiplication factor P) of the redundant components. It is mainly this trade-off that enabled the achievement of equivalent levels of PFD_{avg} with better STR and LCC. This interesting result may be a general design lesson, but it requires further research in order to be validated as such. It can be finally said, to conclude, that the multi-objective optimization of safety systems is therefore highly enhanced by the inclusion of the voting level M as an additional decision variable.

CHAPTER 7

Concluding remarks

7.1. SCOPE OF THE THESIS

This thesis has addressed the multi-objective optimization of Safety Instrumented Systems in compliance with the Standard IEC 61508. The project has made an integration of dependability and Lifecycle Cost modelling with multi-objective optimization using Genetic Algorithms. The focus is on systems working in a low-demand mode of operation (opposite to continuous operation mode systems like process-control systems). Therefore, the metric used for safety integrity is the average Probability of Failure on Demand. Although IEC 61508 does not make any requirement for performance in terms of spurious operation, this is a fundamental aspect of SIS. It certainly has an impact on the LCC of the system and the confidence that the operator places on the system. Therefore, the Spurious Trip Rate is the metric used for measuring reliability of the system. Thus, PFD_{avg} and STR were the two dependability objectives that were optimized. The third objective was the LCC, comprising procurement, operation and risk costs. The cost of risk included production losses by spurious operations and the costs of potential hazards. Therefore, the multi-objective optimization of these three objectives permitted achievement of SIS with a good balance of safety, reliability and costs.

The project concentrated on the study of dependability modelling at component and subsystem level. Dependability modelling at system level was made with Fault Tree Analysis, since the method is versatile enough to model large complex systems without the model becoming too complex itself, as can happen with other methods (i.e. Markov chains). In order to be able to accommodate the changes in architecture design and test policies during the optimization two strategies have been followed: One, the usage of house events that allows the switch on and off of different branches of the fault tree model to cover all potential designs. Secondly, the solution of basic events with different approaches according to the level of complexity of the problem being handled. For this purpose a new PFD(t) model was developed motivated by the need of accommodating the changes in testing policy. The fault tree is modularized by subsystems that are solved by the PFD(t) model.

The second stage for solution of the optimization problem was the integration of the models into the multi-objective optimization. Genetic Algorithms have been selected because they are a powerful tool suitable for ill-structured problems; such as those with non-linearities, discontinuities, mixed variables and even non-explicit functions. These are usual characteristics

of reliability optimization problems. Genetic Algorithms permit truly multi-objective optimization: all the objectives are given the same importance and optimized at the same time. This gives as a result a set of Pareto-optimal solutions. In this set no solution is better than any other: all of them are optimal and only differ in the trade-offs between their objectives. Thus, the decision maker (usually a human) has several options from which to pick the most appropriate solution.

Four major strategies for optimization of SIS, each corresponding to one chapter, have been approached:

1. Optimization of system design with redundancy allocation and component selection (reliability allocation at discrete steps) with parallel redundancies.
2. Optimization of system design with diverse redundancy. The same as the previous case, but now the optimizer has the option to combine different technologically-diverse components into the redundant subsystems.
3. Optimization of the safety system's testing policies. This included the Test Interval and Test Strategy (i.e. the scheduling of the test events of the components with respect to one another) at subsystem level (TI+TS) for parallel systems.
4. Optimization of system design and test with MooN voting architectures. In this, the optimizer can choose the number of redundant components N plus the number of voting components M . In the testing case it includes the TI+TS optimization together with the voting level M .

In this thesis the search was not implicitly constrained with the aim of developing the ability of the optimizer to explore the whole search space and the relationship among the different objectives along the entire Pareto-optimal front. Therefore, the case studies included did not present the desired level of PFD_{avg} as a hard constraint. This is proposed as a future research opportunity in the Future Work section below.

7.2. OPTIMIZATION OF DESIGN WITH PARALLEL REDUNDANCIES

The first optimization case, parallel redundancy allocation and component selection plus TIs, demonstrated that the implementation of the safety system was cost effective since reducing the PFD_{avg} reduces also the LCC in the area of highest PFD_{avg} . The convenience of using multi-objective optimization was also verified because the set of optimal solutions were better than any other solution in the search space. The relationship between the two dependability measures, PFD_{avg} and STR, and LCC was also analyzed. It was confirmed that PFD_{avg} and STR are valid metrics that reflect the reality of the two performance aspects of SIS, safety integrity

and reliability. They were optimized against total unavailability (encompassing the effects of both safe and dangerous failures in one single metric). It was seen that the inclusion of this objective provided only limited additional information that PFD_{avg} and STR did not already provide. The overall unavailability was largely dominated by the PFD_{avg} . Its introduction did not make clearer the information already provided by the PFD_{avg} and STR about the system's performance. It did not help to simplify the decision-making process either, conversely making it rather more complex. Therefore, it was omitted in subsequent optimization analyses.

This first analysis showed that, in the optimal set, the PFD_{avg} is usually in conflict with the LCC. There were, however, three regions:

1. A region of "best" solutions, with the lowest LLC.
2. A region of fair trade-offs where reducing the PFD_{avg} increments the LCC. However, any solution in this set was still better than any other solution in the rest of the search space.
3. A saturation point, after which marginal reduction of the PFD_{avg} required very large increments of the LCC. This would be justifiable only for compliance of some specific safety requirement.

The PFD_{avg} and the STR were generally conflicting. However, there were some regions where reductions of the PFD_{avg} did not require considerable increments in STR. The STR and LCC showed a harmonious relationship.

The application case showed that the optimizer preferred, for lower levels of PFD_{avg} , components with high integrity specifications (low failure rates and high diagnostic coverage, see Section 6.4.3). In the search of lower PFD_{avg} , large changes were made to the subsystems with low-reliability components, showing in their variables low variability and a trend towards the best possible specification. As a second strategy, the subsystems with high-reliability components provided many more opportunities for fine trade-offs.

The optimal solutions of this first optimization case depended heavily on the lowest possible β factor and TI. This, first of all, indicated that CCF was a dominant issue in all the three objectives, and it was necessary to approach it. Testing was assumed to be perfect, without making any special consideration of adverse effects such as unavailability through downtime and introduction of spurious trips. Therefore, it was necessary to explore separately the optimization of testing from the optimization of design in order to make a deeper analysis.

The optimization case was thus fixed with a goal to achieve SIL 3 and a specific maximum level of PFD_{avg} . Three cases regarding the treatment of the initial population in the optimization case

showed that feeding the optimal set as an initial population in consecutive runs gives more optimal results than starting with a new random initial population. A third optimization case fed the optimal set but only in the region of interest when seeking a specific goal (e.g. SIL 3), and this guided the search giving more optimal solutions in the area of interest than simply initiating with a new random population.

7.3. OPTIMIZATION OF DESIGN WITH DIVERSE REDUNDANCY

The second optimization strategy was the optimization of SIS design with diversity at subsystem level. This was the first study that approached diversity in optimization, and several revealing findings were made. First of all, two optimal sets from two different cases were compared, one allowing diversity and the second not allowing it. The optimal set of diverse solutions in general dominated the set of non-diverse solutions. This means that introducing diversity allowed the achievement of better trade-offs.

As a result of the optimization with diversity, the set of optimal solutions contained a large proportion of solutions with diversity in at least one of the subsystems. The solutions in the optimal set with no diversity were only achieving SIL 1. For higher SIL levels (2 to 4), diversity was always in place. A Diversity Index was formulated to represent the relative level of diversity in the system also considering the level of redundancy. The DI is a first attempt to quantify diversity, which provides a useful guideline for the analysis.

Changes in the DI amongst the solutions of the optimal set showed a general trend in which the increment of the DI reduced the PFD_{avg} . Although the general trend was to increment the LCC with an increase of the DI, in a few specific cases this also reduced the LCC, more commonly when SIL was 1-3 ($PFD_{avg} > 1 \times 10^{-4}$). This could be seen in cases where redundancy was reduced and diversity increased simultaneously, improving the STR (when reducing its CCF) but not affecting considerably the LCC. On the other hand, most of the improvements in PFD_{avg} through diversity were in the area of highest integrity (SIL 4) where the cost of marginal reduction of PFD was very high. This is only justified if these high integrity levels are an actual requirement. The optimizer of the case approached gave a large percentage of optimal solutions in this area (SIL 4). A convenient exercise would be to guide a new optimization process towards a specific area of the search space (e.g. SIL 3), and to find if the improvements of PFD_{avg} are less costly.

The results obtained in this chapter regarding the introduction of diversity and the reduction of PFD_{avg} and STR as a consequence were previously expected, and it has been reassuring to see

them as evidence. No previous study has approached the introduction of diversity in the optimization process and tried to quantify its effects. This is a contribution of this work.

It is evident that the presence of CCF is a significant issue that can reduce considerably systems dependability. Quantification of CCF is not a trivial matter. Several sophisticated methods are available for this purpose. However, in the opinion of the author, the only method that can be traceable to actual plant conditions without very complex treatment is the β factor. More complex and accurate methods could be applied if a reliable and feasible procedure to estimate the multiple parameters was available.

7.4. OPTIMIZATION OF TESTING POLICIES

In order to approach the optimization of testing policies a new model for PFD(t) was developed. The model has demonstrated to be able to accommodate different testing policies, i.e. Test Intervals and Test Strategies. Since it was intended for optimization of SIS, it was empowered with the capability to include Common Cause Failure and diagnostic coverage. At the first stage it was developed for parallel redundancies. CCF was modelled as a single additional component, as it is usually done in combinatorial models (e.g. Fault Tree Analysis).

A case study for application of the PFD(t) model was carried out. It was verified that through proof testing it was possible to enhance considerably the safety integrity of the system without actually having to modify its design; i.e. lower Test Intervals and test staggering had a beneficial effect on the PFD_{avg}. On the other hand, a sensitivity analysis showed that CCF may indeed become a dominant factor in the PFD_{avg}. This is especially true in safety systems, where components with very low failure rates are generally used. An importance analysis demonstrated that in these cases the CCF becomes the dominant cut set on the redundant arrangement's fault tree. The time dependent graphs showed that when there was no CCF ($\beta=0$) the PFD_{avg} was significantly lower than when having a small CCF ($\beta=1$): the total PFD(t) had even the same shape of the CCF component of the PFD (PFD_{CCF}). The increment was of 400%. The influence of the diagnostic coverage was also analyzed. Although this was not as dramatic as the influence of CCF, it was certainly important. For instance, a change from $\epsilon=20\%$ to $\epsilon=50\%$ could help the system to achieve one SIL level higher.

Following the practical approach of the thesis, an investigation into current standards and practices of proof testing in the process industry was carried out in order to detect the opportunities for optimization. It was found that testing at a subsystem level is a widely accepted practice, and that the frequency of testing is not regulated whatsoever, only some loose

guidance is given by some standards. In addition, regarding test strategy no guidance was found. Therefore, testing at subsystem level was explored in the optimization cases.

The optimization of testing policies included TI+TS. The TS has been determined by the time to first test of the first component T_{P1} and a staggering factor K , which determined the level of distribution of the test events of the components of a subsystem along the TI. The fault tree of the system was split into modules, each module being a subsystem. These modules were solved by the PFD(t) model. Regarding the quantification of STR, the adverse effects of the testing activity were included as test-induced spurious trips. In this way a better balance of the dependability objectives and the LCC was sought when including both beneficial and negative effects of testing.

The NSGA-II was used for the implementation of the optimization algorithm. The addition of controlled elitism was the tuning change that provided the most significant enhancement in terms of diversity of solutions along the Pareto-optimal front and new solutions at both its extremes. It was verified that the optimization algorithm provided a set of solutions that were superior to all solutions of the initial population set. Thus, its effectiveness was demonstrated.

Comparing two different optimization cases it was possible to verify the crucial importance of including the adverse effects of testing on the STR. An optimization case without this consideration showed that increasing the testing frequency and staggering reduced the PFD_{avg} without affecting the STR, and therefore the costs by safe failure. However, when including the mentioned adverse effects in the optimization, the conflictive relationship PFD_{avg} -STR becomes evident again. When increasing the testing activity for reduction of PFD the STR increased, also raising the LCC considerably. A saturation point in the Pareto front even appeared after which further decrements of the PFD had very high costs in terms of both STR and LCC. Thus two conclusions arise: First of all, not considering the adverse effects of testing can very likely lead to overoptimistic results. Secondly, with this consideration the relationship amongst the three objectives, PFD_{avg} , STR and LCC are very similar for both optimization of testing policies and optimization of system design, with three regions in the Pareto front: one region of better trade-offs where diminishing the PFD_{avg} is not very costly, second where this cost is more prominent and a third of saturation.

Regarding the test policies obtained through the optimization process it has been seen that the reliability specifications of the components largely determine the frequency and staggering of testing. The lower the reliability specifications the higher the required testing activity. Even the test frequency of the subsystems with lowest reliability (and lower redundancy) shaped the

improvements of the PFD_{avg} (e.g. the graph of the PFD_{avg} the system had the same shape of the graph of the TI of the lowest-reliability subsystem). In order to achieve lower PFD_{avg} levels the low-reliability subsystems have to be tested earlier in the operational life, more frequently and more staggered. The two latter approaches, however, have been seen to increase the STR. Therefore, to fully stagger the test to improve system safety, as it is commonly believed, disregarding secondary negative effects is not advisable. It is evident that devices with high reliability specifications require a higher initial expense (by acquisition costs) but in the long run (system operational life) they are more cost-effective.

7.5. OPTIMIZATION OF DESIGN AND TEST WITH MooN VOTING ARCHITECTURES

In the last chapter of the thesis the previous work was extended for optimization of system design and testing policies including MooN voting redundant architectures. This required reformulation of the $PFD(t)$ and STR models and revision of the construction of fault trees in order to accommodate the voting systems. The basis of the analysis has been to determine the behaviour of the system during normal operation and how this is reconfigured during test (and repair), given that one of the components is taken out of service. This depends on the operational philosophy of the system. For this study the philosophy was formulated to be "continue production during testing with degraded operation in the safest possible way". The component under test is bypassed during the test, and the mode of this bypass (open or close) depends upon the operation philosophy. It is this bypass that determines the reconfiguration of the system. A large part of the analysis has been focussed on the changes in CCF. It has been found that, contrary to previous conclusions, given that the component under test is not failed, its CCF can still affect the other components and drag them towards simultaneous failure.

It has been seen that the $PFD(t)$ can be conveniently approximated by quantifying the first-order cut sets plus the next lowest-order combinations of simultaneous independent failures. Thus, the $PFD(t)$ model originally formulated in a previous chapter was only slightly modified. The CCF contribution to the $PFD(t)$ is still modelled as a single additional component, shaped by the specific modification factor C_{MooN} . The quantification of failure rates and how the combinations in the fault trees were created are the main modifications in the modelling strategy.

In order to meet the operational philosophy the bypass for 1ooN systems was set to be closed for 1ooN (parallel) systems and open for MooN where $M > 1$. When implementing a bypass, this has a changing effect on the dependability measures of the systems. This effect is opposite in the PFD and the STR.

Based on first-order cut sets, it has been seen that PFD of 1ooN systems increases during test. On the contrary, for MooN systems (with $1 < M < N$) it remains unchanged, and for NooN it actually decreases. A case study demonstrated that with a fixed number of redundant components N , increasing M raises the PFD. On the other hand, with fixed M , increasing N reduces PFD.

Regarding the STR model, quantification of the STR by internal failures was approximated in the same fashion as the PFD: quantifying first-order cut sets plus the lowest-order combinations of simultaneous independent failures. The probabilistic analysis revealed that the STR of a 1ooN system decreases during test, and for a MooN system ($M > 1$) it actually increases.

Modifying the previous assumption, it was considered that proof testing actually has some coverage over safe failures. This has been included in the modelling of STR as for the negative effects previously addressed. The test-induced increment in STR is a contribution that takes place only during test; thus duration of the test (and repair) time matters.

The case study showed that the effect of increasing either M or N with the other variable fixed is the opposite in STR than in PFD_{avg} : Increasing M (at fixed N) reduces the STR. On the contrary, increasing M (at fixed N) raises the PFD_{avg} . Remember that the total STR is the sum of STR by internal failures and test-induced. It was seen, thus, that the length of the Test Interval affects the influence of STR_{test} over the total STR. The larger the test interval the less significant the test-induced contribution to the total STR (per year), and thus the more equivalent to the STR related to internal failures.

A design optimization case study compared two different cases: one of redundancy allocation and component selection plus the option to change the voting level M . The second case only had parallel architectures (fixed $M=1$). The main finding is that comparing both Pareto-optimal fronts the optimization with MooN architectures case dominated the one with only parallel subsystems. This means better solutions: For pairs of solutions with the same PFD_{avg} , the optimization with MooN gave lower values of STR and LCC.

The optimal front of the optimization with MooN had two regions regarding the relationship PFD_{avg} -LCC: One of harmonious relationship (lowest PFD_{avg} -values end of the graph), and another one with the two objectives in conflict, giving a U-shaped curve. There was in this case no evident saturation point as in previous optimization cases.

The optimizer consistently selected for the optimal set of solutions devices with the best reliability specifications, confirming the conclusions drawn above. The main reductions of PFD_{avg} along the optimal front were made by incrementing the redundancy of the only subsystem with only parallel architecture. It was possible to see that the design of this subsystem had a prominent influence over the PFD_{avg} shape. Additional reductions were achieved by adjusting the redundancy of the PT and TT subsystems, towards a higher redundancy N and lower voting M for lower values of PFD_{avg} . This in contrast increased the STR and to some extent the LCC. Therefore, it can be said that the two ends of the front correspond to, at one extreme designs with $NooN$ subsystems and high PFD_{avg} and low STR, at the opposite extreme designs with $1ooN$ (parallel) subsystems with low PFD_{avg} and high STR.

It is clear that introducing the voting level M as a variable increased significantly the option for new trade-offs. Trading M and N of the subsystems allowed a new balance of the LCC, giving the U-shaped optimal front. One half of the front has the PFD_{avg} in harmony with the LCC: both decreasing along the front (with increasing STR). The second half of this relationship is switched to conflictive: decreasing PFD_{avg} with increasing LCC (and STR). The first half has the lowest STR, while the second half has a steeper increasing STR. This leads one to think that there is a point in which the STR becomes so prominent that it starts dominating the LCC. Thus, further reduction of PFD_{avg} (that is in conflict with the STR) leads to an increment in LCC. Another conclusion is that systems with a high level of voting (low N/M ratio) achieve lower values of STR with a general positive impact on the LCC. On the contrary, parallel systems ($M=1$) reduce the PFD_{avg} but increase the STR and, with this, the LCC.

The second case study addressed optimization of testing policies plus the additional option to vary the voting level M of some subsystems. Two optimization cases were compared again, one with the voting level M of some subsystems as variables and another with fixed $M=1$ (i.e. parallel redundancies). Again, the optimization of testing policies with variable M gave results that dominated the ones with fixed M . At same levels of PFD_{avg} lower values for STR and LCC were achieved.

The optimization of testing policies plus variable M gave an optimal front similar to the design optimization case: split into two regions, one of harmony between PFD_{avg} and LCC and another of conflict.

In the optimization of testing policies with only parallel architectures the PFD_{avg} was largely influenced by increasing the testing activity on the FC subsystem (the one with lowest reliability specifications), followed by increment of testing in the PT and TT subsystems (same as the

results in the previous case study). The PFD_{avg} is consistently in conflict with the STR and the LCC.

In contrast, optimization with the additional voting level M gave for the same levels of PFD_{avg} lower values of LCC and STR (compared with the parallel-only case). This is considerable regarding STR values. However, the Test Intervals were lower (than in optimization of test in the simple parallel system) and time to first test T_{P1} values remained low; i.e. higher testing frequency. In contrast, the level of staggering (determined by the multiplication factor P) has lower and more changing values. Thus, the trend of the TS is not towards full staggering (uniform distribution of the test events along the TI). The changes in the variable M are the same as in the design optimization case: Higher values of M for lower STR and towards $M=1$ (parallel redundancy) for lower PFD_{avg} . However, there are areas where the variable has fast-changing values. In this, both the PFD_{avg} and STR alternate in opposite directions. It could be said that trading between the Test Interval and test staggering P and the voting level M (rather than just varying the testing variables) allows achievement of low values of PFD_{avg} with lower LCC and STR than in the simple-parallel case. Thus, varying the level of staggering, rather than just fully staggering the test events, combined with changes in TI and M can bring benefits. Notice, however, that very low levels of staggering (towards sequential test) increments significantly the LCC.

It can be said, to conclude, that the addition of M as a variable provided many more trade-offs for the optimal front. It also extended the search space and pushed the optimizer to find solutions in a region where the parallel-only optimization case did not. It also allowed the U-shaped optimal front, which can be explained by the changing influence of the STR as above.

7.6. IMPLEMENTATION OF THE GENETIC ALGORITHM

The implementation of the Genetic Algorithm has been made using two leading Genetic Algorithms: the Fonseca & Fleming MOGA and the NSGA-II based on Matlab[®].

The initial set of parameters was made based on recommendations of previous studies on theory of GAs and applications to RAMS+C optimization detailed in Chapter 2. Some generalizations were found that can be used as initial tuning for future studies:

- Coding can be binary or integer for integer variables (such as redundancy level, type of component, etc.)
- A population larger than 100 did not show any enhancement in the search. Thus a population ≤ 100 is convenient.

- Generations: Even for a very large search space (e.g. in the order of 1×10^{30}) a combination of number of generations and population size to give around a total of 5000 evaluations per run seemed to satisfy the balance of proximity and diversity of the Pareto-optimal front (i.e. Generations x Population = 5000).
- Crossover: Single Point Crossover at 0.7 probabilities works well for binary codes. For integer coding, blending algorithms worked well at 0.7 probability (algorithm given by Mulenbein & Schlierkamp-Voosen (1993), see Eq. (2.9))
- Mutation: Bit flipping works well for both coding schemes. A low mutation probability ≤ 0.1 is the best. Some authors recommend a rate of $1/\text{chromosome-length}$.

Tuning for the optimization cases of this thesis was made comparing qualitatively the spread of solutions along the obtained Pareto-optimal set (diversity), the appearance of new solutions at both end extremes of the front and the size of the set. With the advent of second-generation Genetic Algorithms the emphasis of research changed from simplicity to efficiency (as described by Coello-Coello (2006)). This has brought, apart from more efficient GAs, the formulation of new performance measures to quantify the performance of the algorithms. This is an opportunity for future research to improve the tuning process of the GAs.

The first two optimization cases were implemented using the Fonseca & Fleming MOGA, which demonstrated good results. For the second half of optimization cases the implementation was switched to NSGA-II in order to improve the optimizer performance and update it to the state-of-the-art. It would be expected that the implementation of the optimization cases of Chapters 3 and 4 (those done with MOGA) would provide similar results with the NSGA-II, although this would be possibly improved in terms better distributed solutions closer to the real Pareto-optimal front (which is unknown).

The implementation of optimization by GA produced valuable results when confirming relationships among the objectives (PFD_{avg} , STR and LCC) already expected. This is a positive outcome that validates the implementation. The fact GAs produce a pool of optimal solutions has also made possible to visualize, with the aid of graphs, relationships between decision variables and objectives in a large range of solutions, becoming useful tool for system analysis.

7.7. VISUALIZATION OF OPTIMAL SOLUTIONS

Visualization of results given by multi-objective optimization problems can become an issue due to the large amount of information produced. There is first the visualization of optimal solutions, the illustration of which becomes difficult when there are more than two objective

functions involved. In addition, the presentation of the resulting combinations of decision variables can be difficult to analyze in both tabular and graphic form. Thus, the visualization techniques implemented in this work sought to implement a tool to provide the decision maker with all the information needed to analyze and decide. Three techniques have been used:

1. Graphs of the objective functions in three and two-dimensional spaces. This shows the relation of the three objective functions firstly in a three-dimensional graph depicting the general performance of the optimal set. Then the objective functions are plotted in pairs. This facilitates the analysis between the pairs PFD_{avg} -STR, PFD_{avg} -LCC and STR-LCC. Also, they are very useful to compare optimal sets from different optimization cases. This representation is very similar to the Scatter-Plot Matrix method (as described in Deb (2001)), where all pair combinations are plotted. However, in this work each combination is only plotted once to avoid duplications (e.g. PFD_{avg} -LCC and LCC- PFD_{avg}).
2. The Parallel Coordinates graph. This is similar to the Value Path Method (see Deb, 2001). The method of parallel coordinates is the representation of each non-dominated solution from the Pareto-optimal set in two coordinates, with each point in the X-axis being one of the objectives, plotted against the normalized objectives in the Y-axis. The points are united to by a line, each line representing one solution. This is useful to identify conflicting relations when lines are crossing and harmonious when they are concurrent. Therefore, it is convenient for visualization of trade-offs between objectives. However, the graph requires an additional effort to plot it, since it needs normalization of the objectives. The author considers that the graph is useful when the optimization involves many objectives (e.g. >5). However, for a three-objective optimization the graph does not present a clear advantage: The user must perform the reverse conversion of the normalized values and scan amongst many criss-crossing lines that agglomerate too much information and do not highlight specific solutions. This makes it difficult to read and interpret. For a more efficient use of this graph a more sophisticated, maybe interactive, graphic interface is required.
3. A group of (two-dimensional) graphs that shows the values of either the objective functions or the decision variables against all solutions of the optimal set. This is a novel contribution of this work. Each point of the x-axis corresponds to one single optimal solution, and the multi-scale Y-axis represents the values of the objective functions/decision variables. Thus at least two graphs are needed, one with the objective functions and one with the decision variables. In the optimization cases one graph was used for each subsystem. All the graphs are aligned vertically, so that it is possible to visualize all the information, values of objectives and variables, related to each solution. These graphs have been demonstrated to

be very useful for analyzing the influence of decision variables in the objectives and to determine the different influences per subsystem. The author has found these graphs easier to read and interpret than the Parallel Coordinates method and more informative. They were very convenient for optimization problems of the size presented here. In contrast, although they could be extended for a slightly larger number of objective functions (e.g. up to 6), it is considered that they would become too big and complex for visualization of results of optimization cases for very large problems, with many objectives and many decision variables.

7.8. IMPORTANCE OF MULTI-OBJECTIVE OPTIMIZATION OF SIS AND APPLICABILITY OF THE METHODOLOGY

This thesis presents an initial endeavour to integrate a methodology for multi-objective optimization of Safety Instrumented Systems. It analyzes cases of a single safety function. It has been seen that the optimization of safety functions with only some few subsystems can involve many decision variables resulting in very large search spaces (between the order of 1×10^3 up to 1×10^{30}). Since life-size SIS usually have more than one safety function, and frequently more than 10, the complexity of these optimization cases would grow considerable. It is clear that to conduct an exhaustive search to find the optimal solutions is impracticable for these cases. This brings the necessity of multi-objective optimization. The effectiveness of multi-objective optimization, on the other hand, has been verified given that in every case the optimal set of solutions obtained is superior to any solution of the initial population. In addition, the search during the optimization process can be constrained so to give only feasible solutions and guided towards the satisfaction or pre-established goals, as can be seen in Chapter 3.

Multi-objective optimization is required when at least some of the objectives are in conflict. It has been seen that in some areas of the search space the PFD_{avg} and LCC can be reduced simultaneously, which indicates that the implementation of the SIS is cost-effective. Nevertheless, the relationship between objectives in the Pareto-optimal front is quite different. First of all, the system PFD_{avg} is consistently in conflict with the STR in the optimal set resulting from both design and testing optimization. In addition, the relationship between LCC and PFD_{avg} is mostly conflictive, and between LCC and STR mostly harmonic (see Figs. 3.14, 5.20 and 6.26). This means that the higher the LCC, the lower the PFD_{avg} and the higher the STR. These relationships are prevalently true, but not true for all cases. For some optimization cases involving diverse redundancy or optimization with MooN voting systems these relationships are inverted in the higher- PFD_{avg} side of the optimal set; i.e. lower LCC goes alongside lower PFD_{avg} and higher STR. For example, for optimization with MooN voting

redundancies, solutions with not very low PFD_{avg} (e.g. $\geq 1 \times 10^{-3}$ or $\leq \text{SIL } 2$) this relation is harmonious. For these optimal sets the plot of LCC values takes a U shape with respect to PFD_{avg} and STR (see Figs. 4.11, 6.20 and 6.26).

The relationships described above can be partly explained by the influence of both risk costs (i.e. those related to dangerous and safe failures; Eq. 4.19) on the LCC. The cost of hazards is determined by the cost of a potential catastrophic accident and its estimated frequency. However this is significantly reduced when multiplied by the PFD_{avg} of the SIS. In this case, the costs of production loss by spurious trips, determined by the STR, become dominant over the LCC. Therefore, since PFD_{avg} and STR are conflictive measures, at lower PFD_{avg} one gets higher STR and, as a consequence, higher LCC. This is the mostly dominant case. However, this balance has been altered when introducing diverse redundancy in the optimization of design, and significantly when introducing voting systems in the optimization of design and testing, although only in the section of the PFD_{avg} graph with higher values (this can be seen clearly in Fig. 6.20). It can be thought that in this section the costs related to dangerous failures are not so significantly reduced, and therefore become more prominent than those originated by the STR. Thus, the lower the PFD_{avg} the lower the LCC (and the higher the STR). There is a point around which this relationship is inverted (originating the LCC U-shaped graph), and the effects of STR become the dominant factor in the LCC, as explained above. In correspondence to PFD_{avg} the value of this inversion point varies. In the optimization cases studied here it lies between 1×10^{-3} and 1×10^{-2} .

The complexity in the relationship between objectives described above, together with the large search space originated by the multiple decision variables involved in both design and testing of SIS, reinforces the importance of multi-objective optimization.

The results obtained through the optimization process and the visualization tools used have allowed an analysis of the relations between decision variables and the objective functions. This made also possible to identify trends and the level of influence per subsystem (for instance, the Final Control subsystem was identified as the weakest arrangement of the SIS). This has provided an additional insight into the analysis of SIS. Therefore, the modelling and optimization method presented here has turned into an effective analysis tool.

The methodology developed for the dependability measures allows hierarchical modelling of Safety Instrumented Systems, from component level (where the different failure modes are defined) and subsystem level (integrated as redundant arrangements with quantification of CCF) to system level. The power of the model resides in its capacity of quantifying explicitly different

failure modes, the capability of modelling time-dependent changing factors (in the PFD) and the flexibility to accommodate changes in both design and testing policies.

The models developed here are based on quantification of CCF by the β factor model. The rationale behind this is that it is the most widely applied method due to its simplicity and traceability to real design and operating conditions. Therefore, the application of the method is only enabled by the β factor model. This is also the basis of its strength as a practical method.

There are limitations in the method formulated in this thesis as is normal for every methodology that intends to be applicable in practise. The method has been developed to comply with IEC 61508, and this gives its strength as a practical approach. Although it does not intend to be a universal method applicable to every existing standard, it may be easily applicable under the framework of other standards as long as the assumptions made here are met. This is especially true for low-demand systems. For continuous operating systems further study is required to expand its applicability.

The method has been approached having in mind a conceptual design stage. At this stage some limitations are that some data and the full architecture definition may not be available, and relevant assumptions would have to be made. These assumptions can be revisited and reviewed to during the detailed design stage. In Safety Instrumented Systems the requirements during the conceptual design and detailed design stages are not considerably different. The main difference is the final definition of design and operating conditions. Therefore, the method is still applicable if the assumptions made are still valid. It can thus be used as a design and decision making tool for Safety Instrumented Systems and similar systems.

The limitation of sufficient detailed data certainly exists. Especially dependability data for components at the level required by IEC 61508. The standard, however, is becoming widely used in several sectors, and the availability of data is becoming common place. As a result of its widespread used, generic databases have been appearing recently dedicated to SIS (i.e. Hauge et al., 2006a; Exida, 2008). These can be a source of data to be used to substitute for real meaningful data under the correct assumptions.

Another limitation for applicability of the method is the need for tuning the parameters of the Genetic Algorithm. However, good guidance of initial tuning can be found in the literature referenced in Section 2.4 and Appendix B, and also in the tuning schemes presented here. Also, as it was seen at the end of Section 2.2.2, GAs can still be robust to the lack of perfection in some parameters and still deliver good results.

Every methodology is subject to limitations and with particular assumptions are still applicable. All the mentioned limitations are opportunities of further research, and they are proposed in Section 10 Future work as such. The methodology developed here is a powerful tool for decision making, and the opportunities of further expansion are certainly promising.

7.9. MAIN ACHIEVEMENTS OF THE THESIS

The main achievements accomplished through the work described in this thesis are detailed below:

- The prime achievement has been to provide a deep analysis of safety system dependability modelling with sufficient detail for compliance with IEC 61508 requirements and its integration with multi-objective optimization.
- The development of a PFD(t) model for MooN architectures (that includes parallel redundancies).
- The development of a STR model that includes spurious trips from internal failures and by test-adverse effects.
- To devise a Lifecycle Cost model that includes both procurement and operation along the complete system's operational life, and that considers the effects of both safe and dangerous failures of the system.
- The approach of safety system optimization addressing important questions that are frequently overlooked:
 - Common Cause Failure quantification and its mitigation by diverse redundancy.
 - Optimization of MooN redundancies with different levels of voting M.
- A comprehensive study of proof testing in the process industry and its effects, positive and negative, in system dependability.
- Integration of dependability modelling at component level with Fault Tree Analysis for modelling at system level, and capable of accommodating variable conditions suitable for optimization.
- Integration of safety system dependability modelling with application of leading multi-objective optimization Genetic Algorithms.
- An analysis of application of two leading Genetic Algorithms for safety system optimization that open the door for further research into tuning and parameterization.
- Formulation of a novel graphic technique for analysis of results. This has provided, as an additional benefit, a tool for comparison of different optimization cases and for analysis of system design and testing policies with relation to system performance in the dependability and cost measures.

7.10. FUTURE WORK

The avenues of further research that have been opened up through this work are numerous, stretching to the field of system modelling at component and system levels, tuning of Genetic Algorithms, integration of modelling and optimization algorithms to encompass diverse optimization strategies, scaling up the entire methodology and extrapolation of the methodology to other types of safety-critical systems. The avenues for future research can be enumerated as follows:

1. The most imposing task is the integration of all the optimization strategies into one single optimization procedure. Specifically integration of optimization with diversity into design optimization of MooN voting architectures. This will increase considerably the complexity of the fault trees and their solution. One option to explore is the use of Binary Decision for synthesis of fault trees, such as applied by Andrews' research group (Pattison & Andrews, 1999; Andrews & Bartlett, 2003; Borisevic & Bartlett, 2007a, 2007b; and Riauke & Bartlett, 2008).
2. The second step would be the integration of optimization of system design and testing polices together. This could be done as a double nested loop, the other loop optimizing design and the inner loop testing policies. A similar exercise for optimization of test intervals and strategies was made by Martorell et al. (2006).
3. Scaling the method up for optimization of life-size SIS with several safety functions (e.g. ≥ 10). This will mainly bring forward the problem of growing complexity, and for which a feasible path of research is the application of Binary Decision Diagrams for solutions of fault trees as mentioned above.
4. The method for pruning Fault Trees developed by the Lutz research group (Lu & Lutz, 2002; Dehlinger & Lutz, 2004, 2006) is worthy of investigation for application in the optimization of SIS design. This permits pruning of a Fault Tree that corresponds to a complete Product Family in order to represent a single product member, enabling reuse of fault trees. The method can be investigated in order to determine whether it could be used as an alternative method for adapting the generic fault tree to represent several optional designs during the optimization process (in substitution of the method based on house events used in this work). The intention would be to see whether it provides a more efficient solution, especially for modelling larger systems.

5. The extrapolation of the analysis of modelling and optimization for systems in continuous mode of operation. This would extend the application to safety-critical control systems that cover both aspects, performance of safety-critical control, such as those used in aerospace and marine industry.
6. The effects of human error on both models, PFD_{avg} and STR, is certainly a topic that requires further research. In Sections 5.8 and 6.35 a first approach to include human error in the form of test-induced spurious trips has been addressed. It can be said that human error affects construction of fault trees in two ways: Firstly, how the probabilities of failures induced by human errors are quantified, and secondly how the basic events of human error-induced failures are placed in the fault tree hierarchy and how they are associated (by logic gates) within it (this is an important issue in MooN voting systems). In addition, how these basic events are reconfigured during testing of one component is also another issue.
7. Several extensions to the $PFD(t)$ model to include further contributions could be made in the future. One very important aspect is the adverse effects of proof testing over the PFD_{avg} . This can be encompassed basically in imperfect testing and the increment of PFD by human error induced faults during test. Vaurio (1995a) found that an additional constant q in the unavailability model (see Eq. (5.2)) could accommodate several diverse contributions, such as human error. Due to the amount of research that this would require, it has not been included in the scope of this work, although it has been acknowledged and well documented in the literature review of Section 5.1.3. Thus, this can be a good starting point for the investigation.
8. To further investigate the influence of proof testing into STR. Proof testing is primarily aimed at detection of dangerous failures. However, it is clear that a proportion of safe failures must be also revealed during the practise of proof testing; what is more, some safe failures are explicitly targeted. Therefore, further definition of the coverage of proof testing over safe failures needs to be done. The investigation into this and the previous point would allow quantification of the effects of proof testing, both positive and negative, over the two different failure modes, dangerous and safe, and hence the two dependability measures, permitting the achievement of a better balance of loss of safety and loss of production.
9. An additional extension of the $PFD(t)$ model to consider is the probability of failure of the system caused by the demand itself (i.e. the demand, by the onset of a potentially hazardous condition, to the system to actuate causes the system itself to fail, as mentioned in Dudit et

- al. (2006)). This is another contribution that can be accommodated in the term q described by Vaurio (1995a).
10. Another important contribution to explore is the constraint of Allowed Outage Time (AOT). This is a limited time span allowed for restoration of faulty equipment within a safety system. If the equipment fails to be restored within this time, the plant must be brought to a safe operational state (Martorell et al., 1995). This concept is widely used in safety systems of Nuclear Power Plants. Martorell et al. (2002, 2004) included the AOT as a decision variable in the optimization of the technical specifications and maintenance of the HPIS. A similar restriction to the AOT is prescribed in IEC 61508 Part 2, Section 7.6.4 as Mean Time to Restoration (MTTR).
 11. Research into quantification of CCF with couplings between different technologies. It has been seen that the introduction of diversity increased considerably the size of the fault trees. These would become considerable complexity issues; the quantification of CCF between technologies and its introduction as basic event into the fault trees, if dependencies (couplings) were considered between different types of technologies in a diverse redundant arrangement. It therefore would require investigating into a CCF model capable of handling quantification of contributions by different couplings in a way traceable to actual design and plant conditions. This would also imply to research into a combinatorial model capable of handling them efficiently in the changing conditions of the design optimization process (which would increase considerably the modelling complexity, as discussed in point 1 above).
 12. Further refinement of the Diversity Index. It may be convenient to refine the mathematical formulation of the Diversity Index (introduced in Chapter 4), especially to separate the effects of changes in diversity from changes in redundancy levels or changes in the technology of the devices.
 13. The powerful PFD(t) model developed here is convenient for accommodation of many different changing conditions. However, for optimization of design it may be convenient to explore whether some more simplified equations to solve the basic events of fault trees give an adequate approximation. A promising possibility is to investigate the extension and application of the equations developed by Vaurio (1994).
 14. Spurious Trip Rate is a widely used metric as a standard to measure the effects of safe failures on SIS (e.g. IEC, 2003; Hauge et al., 2006a; CCPS, 2007; Lundteigen & Rausand,

- 2008b). However, some other authors have used the probability of safe failure (or spurious activation) as a metric (e.g. Goble, 1998; Lu & Lewis, 2006, 2008). Since compliance with some levels of STR is not a requirement of IEC 61508 it has not received much attention. It is, however, a fundamental measure of SIS performance which has a significant impact on the LCC as has been seen in this thesis. It would be interesting to analyze what would be the benefits of using a time-average measure of system unavailability (due to safe failures) or probability of safe failures instead of the STR. This would also require developing a modelling strategy at component and system level for this probability.
15. The optimization of proof testing with varying the level of voting M showed that there is a trading-off between M and the variables TI and P , where P tends towards non-fully staggered testing. This is an interesting result that opens the possibility of becoming a general design lesson for safety systems. To validate it as such requires further research.
 16. Optimization cases where the $PFD(t)$ was used were computationally expensive due to the simulation involving sampling of values along the entire useful time-life of the system. It is a worthy research avenue to explore the application of numerical methods to make the integration of PFD (rather than simulate it) so the computational cost is lowered.
 17. A comprehensive study of tuning of the different parameters of the GA. This includes measuring changes in the performance of the algorithm with changes in the parameters. Tuning has been made by qualitative comparison of the Pareto-optimal front obtained in terms of the quantity of solutions obtained, uniform distribution (exploitation) and new solutions at both extremes of the front (exploration). Given that in these practical applications the Pareto-optimal front is unknown, some performance measures for proximity and diversity cannot be used. However, an alternative parametric study could be made to quantitatively measure which tuning arrangement gives better results. For example quantifying the average fitness of the Pareto front or using metrics of spread.
 18. Multi-objective optimization with constraint handling by a vector of constraints in order to guide the search towards specific regions in order to meet goals. The optimization cases in this thesis have been implemented with explicit constraints that limit the decision variable space to feasible solutions. This is due to the intention to make it a practical approach. An area to investigate is therefore constraint handling in the implementation of optimization subject to a vector of implicit constraints, in which the search must be guided to be compliant with them. A specific case to explore is when PFD_{avg} is a hard constraint, which is a frequent requirement in the specification of safety systems.

19. Guidance of the search process towards specific goals. A convenient future exercise would be to guide a new optimization run towards a specific area of the search space (e.g. SIL 3), and to find if the improvements of PFD_{avg} are less costly.
20. Specifically in the MOGA a deeper treatment of fitness sharing and mating restriction. Specifically for the Fonseca & Fleming MOGA fitness sharing for reduction of genetic drift and mating restriction against the formation of lethals (see appendix A) are two areas to be further explored.
21. The added controlled elitism in the NSGA-II demonstrated to be a very relevant feature of the implementation in Chapter 5. A study into a procedure for tuning the reduction rate parameter r that determines the controlled elitism, and which is problem dependent, would very likely permit considerable enhancement of the performance of the algorithm.
22. The implementation of the decision maker is an area of current extensive research. It has been seen that the optimizer provides tens of optimal solutions for one single safety function. It could be the case that the optimization of a complete SIS with several safety functions would give hundreds of solutions as part of the optimal set. From there the decision maker must usually select one single solution. Therefore, this is another opportunity for research. As it can be seen in Appendix A, Fonseca & Fleming (1993, 1995a) incorporated a form of decision maker in the algorithm. This may be a starting point.
23. The process of making the final decision for selection of one solution from the optimal set requires analysis of the trade-offs between the system's objectives with a consistent methodology. Despotou et al. (2005) and Despotou & Kelly (2005, 2007) presented a methodology based on the ALARP principle and the Goal Structuring Notation for making justified trade-offs in safety critical systems. Investigation into usage of this methodology to guide the decision maker is thus another research opportunity.
24. Visualization of results with many objectives and variables is always complicated. Further research into application of graphic techniques would be useful. Especially development of an interactive user interface would be very convenient to help in the interpretation of the results.
25. The standard IEC 61508 is currently in the last phases of a new revision. It is uncertain when the new version will be published, although it is probable that this will take place in the first months of 2010. As it was mentioned in Chapter 1, there has been some

controversy about the IEC 61508 requirements for specification of SIL levels, including the name of the PFD_{avg} itself (Dudit (2008), see section 1.4.10 of this thesis) and the architectural constraints (Signoret (2007), Lundteigen & Rausand (2008), Yoshimura & Sato (2008); see section 1.5 herein). It will be then important to review the changes made to the standard and make the necessary adjustments to the methodology presented in this thesis.

REFERENCES

- [1] Aarø P., Bodsberg L., Hokstad P. (1989). Reliability prediction handbook; computer-based process safety systems. SINTEF report, Trondheim, Norway.
- [2] Adamsky R.S. (1991). Evolution of protective systems in the petrochemical industry. ISA Transactions, 1991(4), 27-32.
- [3] Adra S.F. (2007). Improving convergence, diversity and pertinency in multiobjective optimization. PhD Thesis. The University of Sheffield, Sheffield, UK.
- [4] Amari S.V., Pham H., Dill G. (2004). Optimal design of k -out-of- n :G subsystems subjected to imperfect fault-coverage. IEEE Transactions on Reliability, 2004, 53(4), 567-575.
- [5] American Petroleum Institute (2001). API RP 14C Recommended Practice. Analysis, design, installation, and testing of basic surface safety systems for offshore production platforms. 7th ed. Washington D.C., USA.
- [6] Andrews J.D. (1994). Optimal safety system design using fault tree analysis. Procs. of IMechE Part E, Journal of Process Mechanical Engineering 1994;208(E2):123–131.
- [7] Andrews J.D., Bartlett L.M. (2003). Genetic Algorithm Optimisation of a Firewater Deluge System. Quality & Reliability Engineering International 2003;19(1):39-52.
- [8] Andrews J.D., Ericson II C.A. (2000). Fault Tree and Markov analysis applied to various design complexities. In Proc of 18th International System Safety Conference. Texas, USA, 2000.
- [9] Andrews J.D., Moss T.R. (2002). Reliability and Risk Assessment. 2nd ed. Professional Engineering Publishing Limited. London, UK.
- [10] Apostolakis G.E., Bansal P.P. (1977). Effect of Human Error on the availability of periodically inspected redundant systems. IEEE Transactions on Reliability 1977;R-26(3):220-225.
- [11] ARC (1999). Critical Control and Safety Shutdown Systems Strategies Technical report. ARC Advisory Group, USA, July 1999.
- [12] Arulmozhi G. (2002). Exact equation and an algorithm for reliability evaluation of K-out-of-N:G systems. Reliability Engineering and System Safety 2002;78(2):87-91.
- [13] Aviezenis A., Laprie J.C., Randell B. (2000). Fundamental concepts of dependability. 3rd Information survivability Workshop, (ISW-2000), Boston, Massachusetts, October 24-26, 2000.
- [14] Back T, Fogel DB, Michalewicz Z (eds.) (2000). Evolutionary computation I. Basic algorithms and operators. Institute of Physics Publishing. Bristol, UK.
- [15] Bai D.S., Yun W.Y., Chung S.W. (1991). Redundancy optimization of k -out-of- n systems with common cause failures. IEEE Transactions on Reliability 1991;40(1):56-59.
- [16] Ben-Dov Y. (1980). Optimal reliability design of K-out-of-N systems subject to two kinds of failure. Journal of the Operational Research Society 1980;(31):743-748.
- [17] Biernat J. (1990). The alternate approach to the reliability modeling of fault masking systems. Microelectronics and Reliability 1989;30(3):503-506.
- [18] Biernat J. (1994). The effect of compensating fault models on NMR systems reliability. IEEE Transactions on Reliability 1994;43(2):294-300.
- [19] Biernat J. (1995). Discussion of compensating faults in modelling of NMR system reliability. Reliability Engineering and System Safety 1995;47(3):221-228.
- [20] Billings S.A., Zheng G.L. (1995). Radial basis function network configuration using genetic algorithms. Neural Networks 1995;8(6):877-886.
- [21] Billington R., Allan R.N. (1983). Reliability Evaluation of Engineering Systems: Concepts and Techniques. Pitman Books Limited, London, UK.
- [22] Bodsberg L., Hokstad P. (1995). A system approach to reliability and life-cycle cost of process safety-systems. IEEE Transactions on Reliability 1995;44(2):179-186.
- [23] Bodsberg L., Hokstad P. (1997). Transparent reliability model for fault-tolerant safety systems. Reliability Engineering and System Safety 1997;55(1):25-38.
- [24] Borisevic J., Bartlett L.M. (2007a). Safety System Optimisation by Improved Strength Pareto Evolutionary Approach (SPEA2). In: Bartlett L. (ed.) Proceedings of the 17th Advances in Risk and Reliability Technology Symposium, Loughborough, UK, 2007, p. 38-49.
- [25] Borisevic J., Bartlett L.M. (2007b). Genetic Algorithm based Multi-objective Optimization of a Firewater Deluge System. In: Aven A. & Vinnem J.E. (eds), Procs. European Safety and Reliability Conference ESREL '07. Stavanger, Norway 2007, p. 107-114.
- [26] British Petroleum (1994). BP RP32-6 Inspection and testing of In-service Instrumentation. British Petroleum Company, Sunbury, UK.

- [27] Bukowski J.A. (2005). Comparison of techniques for computing PFD average. In Proc of the 51st Annual Reliability and Maintainability Symposium. Alexandria, VA, USA, 2005. p. 590-595.
- [28] Bukowski J.V. (2001). Modeling and analyzing the effects of periodic inspection on the performance of safety-critical systems. *IEEE Transactions on Reliability* 2001;50(3):321-329.
- [29] Bukowski J.V., Goble W.M. (1995). Using Markov models for analysis of programmable electronic systems. *ISA Transactions* 1995;34(2); 193-198.
- [30] Bukowski J.V., Goble W.M. (2001). Defining mean time to failure in a particular failure state in multi failure states systems. *IEEE Transactions on Reliability* 2001;50(2):221-228.
- [31] Cantoni M., Marseguerra M., Zio E. (2000). Genetic Algorithms and Monte Carlo Simulation for Optimal Plant Design. *Reliability Engineering and System Safety* 2000;68(1):29-38.
- [32] CCPS (2000). Guidelines for chemical quantitative risk analysis. 2nd edition. Center for Chemical Process Safety. Wiley Blackwell. New York, USA.
- [33] CCPS (2007). Safe and Reliable Instrumented Protective Systems. Center for Chemical Process Safety. Wiley Interscience. New Jersey, USA.
- [34] Cepin M. (1995). Sequential versus staggered testing towards dynamic PSA. In Procs. of the 2nd Regional Meeting Nuclear Energy in Central Europe. Portoroz, Slovenia, 1995. p. 184-189.
- [35] Cepin M. (2002). Optimization of safety equipment outages improves safety. *Reliability Engineering and System Safety* 2002;77(1):71-80.
- [36] Cepin M., Kozuh M., Mavko B. (1994). Risk impacts associated with surveillance tests. In: Procs 4th TUV Workshop on living PSA, 2-3 May, Hamburg, Germany.
- [37] Cepin M., Mavko B. (1997). Probabilistic safety assessment improves surveillance requirements in technical specifications. *Reliability Engineering and System Safety* 1997;56(1):69-77.
- [38] Chari A.A. (1994). Optimal redundancy of K -out-of- n :G system with two kinds of CCFs. *Microelectronics and Reliability* 1994;34(6):1137-1139.
- [39] Chatterjee P. (1975). Modularization of fault trees: A method to reduce the cost of analysis. *Reliability and fault tree analysis*, SIAM, 1975. p. 101-137.
- [40] Chipperfield A., Fleming P., Polheim H., Fonseca C. (1994). Genetic Algorithm Toolbox User's Guide. Research report 512. University of Sheffield, U.K.
- [41] Coello-Coello C.A. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems. An International Journal* 1999;1(3): 269-308.
- [42] Coello-Coello C.A. (2000). Handling preferences in evolutionary multiobjective optimization: a survey. In *IEEE Neural Networks Council (ed.), Proceedings of the 2000 Congress on Evolutionary Computation CEC 2000*. Vol. 1, IEEE Service Center, Piscataway, New Jersey, USA, pp. 30-37.
- [43] Coello-Coello C.A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering* 2002;191(11):1245 - 1287.
- [44] Coello-Coello C.A. (2006). Evolutionary Multi-Objective optimization: A historical view of the field. *IEEE Computational Intelligence Magazine* 2006;1(1):28-36.
- [45] Coit D.W. (2003). Maximization of system reliability with a choice of redundancy strategies. *IEEE Transactions on Reliability* 2003;52(6):535-543.
- [46] Coit D.W., Liu J. (2000). System reliability optimization with k-out-of-n subsystems. *International Journal of Reliability, Quality and Safety Engineering* 2000;7(2):129-142.
- [47] Coit D.W., Smith A. (1994). Use of a genetic algorithm to optimize a combinatorial reliability design problem. In: Procs. Third IIE Research Conference, 1994, p. 467-472.
- [48] Coit D.W., Smith A. (1996a). Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm. *IEEE Transactions on Reliability* 1996;45(2):254-260.
- [49] Coit D.W., Smith A. (1996b). Penalty guided genetic search for reliability design optimization. *Computers and Industrial Engineering* 1996;30(4):895-904.
- [50] Coit D.W., Smith A. (1996c). Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach. *Computers and Operations Research* 1996;23(6):515-526.
- [51] Courtois P.J., Delsarte P. (2006). On the optimal scheduling of periodic tests and maintenance for reliable redundant components. *Reliability Engineering and System Safety* 2006;91(1): 66-72.
- [52] Damousis I.G., Bakirtzis A.G., Dokopoulos P.S. (2004). A solution to the Unit-Commitment problem using integer-coded genetic algorithm. *IEEE Transactions on Power Systems* 2004;19(2):1165-1172.
- [53] Deb K. (2001). Multiobjective optimization using evolutionary algorithms. John Wiley and Sons LTD. Chichester, UK, 2001.
- [54] Deb K., Agrawal S., Pratab A., Meyarivan T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Marc Schoenauer et al. (ed), Procs. Parallel

- Problem Solving from Nature VI Conference. Paris, France, 2000. Lecture Notes in Computer Science No. 1917. London: Springer-Verlag. P. 849-585.
- [55] Deb K., Agrawal S., Pratab A., Meyarivan T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 2002;6(2):182-197.
- [56] Debb K., Goel T. (2001). Controlled elitist non-dominated sorting genetic algorithms for better convergence. In: Zitzler E. et al. (eds), *Procs. First Int. Conf. Evolutionary Multi-Criterion Optimization 2001*. Zurich, Germany, 2001. Lecture Notes in Computer Science No. 1993. London: Springer-Verlag. P. 67-81.
- [57] Dehlinger J., Lutz R.R. (2004). Software Fault Tree Analysis for Product Lines. In *Proceedings of the 8th IEEE International Symposium on High Assurance Systems Engineering*, Tampa, FL, USA, p. 12-21.
- [58] Dehlinger J., Lutz R.R. (2006). PLFaultCAT: A Product-Line Software Fault Tree Analysis Tool. *Automated Software Engineering* 2006;13(1):169-193.
- [59] Despotou G., McDermid J., Kelly T. (2005). Using scenarios to identify and trade-off dependability objectives in design. In: *Procs. 23rd International System Safety Conference (ISSC)*, San Diego, CA, USA, August 2005. Published by the System Safety Society.
- [60] Despotou G., Kelly T. (2007). An argument-based approach for assessing design alternatives and facilitating trade-offs in critical systems. *Journal of System Safety* 2007;43(2):31-21.
- [61] Dhillon B.S. (1989). *Life Cycle Costing*. Gordon and Breach Science Publishers. Amsterdam, Netherlands.
- [62] Dhillon B.S., Singh C. (1981). *Engineering Reliability. New techniques and applications*. John Wiley & Sons, New York, USA.
- [63] Dudit Y., Chatelet E., Signoret J-P., Thomas P. (1997). Dependability modelling and evaluation by using stochastic Petri nets: Application to two test cases. *Reliability Engineering and System Safety* 1997;55(2):117-124.
- [64] Dudit Y., Innal F., Rauzy A., Signoret J-P. (2008). Probabilistic assessments in relationship with safety integrity levels by using Fault Trees. *Reliability Engineering and System Safety* 2008;93(12):1867-1876.
- [65] Dudit Y., Rauzy A. (1996). A linear-time algorithm to find modules of fault trees. *IEEE Transactions on Reliability* 1996;45(3):422-425.
- [66] Dudit Y., Rauzy A. (2001). New insights into the assessment of k-out-of-n and related systems. *Reliability Engineering and System Safety* 2001;72(3):303-314.
- [67] Dudit Y., Rauzy A., Signoret J-P. (2006). Probabilistic assessment in relationship with Safety Integrity Levels by using fault trees. In: Guede Soares C & Zio E (eds), *Procs. European Safety and Reliability Conference ESREL '06*. Estoril, Portugal 2006, p. 1619-1624.
- [68] Dugan J., Bavuso S., Boyd M. (1992). Dynamic fault tree models for fault tolerant computer systems. *IEEE Transactions on Reliability* 1992;41(3):363-377.
- [69] Elegbede C., Adjallah K. (2003). Availability Allocation to Repairable Systems with Genetic Algorithms: A Multi-objective Formulation. *Reliability Engineering and System Safety* 2003;82(3):319-330.
- [70] Exida (2008). *Safety Equipment Reliability Handbook*. 3rd edition. Vol. 1: Sensors. Vol. 2: Logic Solvers and Interface Modules; Volume 3: Final Elements. Exida LCC, Sellersville, Pennsylvania, USA.
- [71] Exxon (1999). IP-15-7-2 International Practice for Protective Systems. Exxon Research and Engineering Company, Florham Park, N.J. USA.
- [72] Fischer H.D., Piel L. (1999). Diversity in computerized reactor protection systems. *Reliability Engineering and System Safety* 1999;63(1):91-97.
- [73] Fleming K.N. (1975). A reliability model for common mode failure in redundant safety systems. In: *Procs of the 6th Annual Conference on Modeling and Simulation*. General Atomic Report GA-A13284. 23-25 April 1975.
- [74] Fonseca C.M. (1995). *Multiobjective Genetic Algorithms with application to control engineering problems*. PhD thesis. University of Sheffield, Sheffield, UK.
- [75] Fonseca C.M., Fleming P.J. (1993). Genetic Algorithms for multi-objective optimization: Formulation, Discussion and Generalisation. In *Proc. of 5th Int. Conf. on Genetic Algorithms*. San Mateo, CA, USA 1993. p. 416-423.
- [76] Fonseca C.M., Fleming P.J. (1995a). *Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms I: A Unified Formulation*. Research report 564. University of Sheffield, Sheffield, UK.

- [77] Fonseca C.M., Fleming P.J. (1995b). Multiobjective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restriction. In Proc. of 1st Int Conf on genetic algorithms in Engineering Systems: Innovations and Application (GALESIA '95). Stevenage, UK 1995. p. 45–52.
- [78] Fonseca C.M., Fleming P.J. (1998). Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms I: A Unified Formulation. *IEEE Transactions on Systems, Man and Cybernetics Part A: Systems and Humans* 1998;28(1):38-47.
- [79] Frederickson A. (1990). Fault tolerant programmable controllers for safety systems. *ISA Transactions*, 1990, 29(2), 13-16.
- [80] Frederickson T., Beckman L.V. (1991). Comparison of Fault Tolerant Controllers used in safety applications. *ISA transactions* 1991;30(4):97-106.
- [81] FSC (1991). OpenFTA Release 1.0. Formal Safety construction Limited, Cardiff, UK. Available online at www.openfta.com. [last accessed 12-October-2008].
- [82] Gen M., Ida K., Taguchi T. (1993). Reliability optimization problems: a novel genetic algorithm approach. Technical report, ISE93-5, Ashikaga Institute of Technology, Ashikaga, Japan.
- [83] Gen M., Kim J.R. (1999). GA-based Reliability design: State-of-the-art Survey. *Computers and Industrial Engineering* 1999;37(1-2):151-155.
- [84] Gen M., Yun Y (2006). Soft Computing Approach for Reliability Optimization; State-of-the-art Survey. *Reliability Engineering and System Safety* 2006;91(9):1008-1026.
- [85] Giuggioli-Busacca P.G., Marseguerra M., Zio E. (2001). Multiobjective Optimization by Genetic Algorithms: Application to Safety Systems. *Reliability Engineering and System Safety* 2001;72(1):59-74.
- [86] Goble W.M. (1998) Control Systems Safety Evaluation & Reliability. 1998. The Instrumentation, Systems and Automation Society. Research Triangle Park, NC, U.S.
- [87] Goble W.M., Bukowski J., Brombacher A.C. (1998). How diagnostic coverage improves safety in programmable electronic systems. *ISA Transactions*, 1998, 36(4), 345-350.
- [88] Goble W.M., Cheddie H. (2005). Safety Instrumented Systems Verification. The Instrumentation, Systems and Automation Society. Research Triangle Park, NC, US.
- [89] Goldberg D.E. (1989). Genetic algorithms in search, optimization and machine learning. Addison Wesley, Reading MA, USA.
- [90] Goldberg D.E., Richardson J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette J.J. (ed.), *Procs. Second International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates, New Jersey, USA, pp. 41-49.
- [91] Gonzalez L., Garcia D., Galvan B.J. (2004). An intrinsic order criterion to evaluate large, complex fault trees. *IEEE Transactions on Reliability* 2004;53(3):297-305.
- [92] Gopal K., Aggarwal K.K., Gupta J.S. (1978). A new approach to reliability optimization in General Modular Redundant Systems. *Microelectronics and Reliability* 1978;18(5):419-422.
- [93] Greiner D., Galvan B., Winter G. (2003). Safety systems optimum design by multicriteria evolutionary algorithms. In: Fonseca C.M. et al. (eds.); *Procs. Evolutionary Multi-criterion Optimization Conference, Faro, Portugal, 2003*. Lecture Notes in Computer Science No. 2632. London: Springer-Verlag, p. 722-736.
- [94] Gruhn P. (1996). The evaluation of safety instrumented systems – tools peer past the hype. *ISA Transactions* 1996;35(1):25-32.
- [95] Gruhn P., Cheddie H. (1998). Safety Shutdown Systems: Design, Analysis and Justification. The Instrumentation, Systems and Automation Society. Research Triangle Park, NC, US.
- [96] Gruhn P., Cheddie H. (2005). Safety Instrumented Systems: Design, Analysis and Justification (2nd edn). The Instrumentation, Systems and Automation Society. Research Triangle Park, NC, US.
- [97] Gulati R., Dugan J.B. (1997). A modular approach for analyzing static and dynamic fault trees. In: *Procs. Annual Reliability and Maintainability Symposium, Philadelphia, USA, 13-16 January 1997*.
- [98] Guo H., Yang X. (2007). A simple reliability block diagram method for safety integrity verification. *Reliability Engineering and System Safety* 2007;92(9):1267-1273.
- [99] Hauge S., Hokstad P., Langseth H., Oien K. (2006a). Reliability Prediction Method for Safety Instrumented Systems. PDS Data Handbook 2006 edition. SINTEF, Norway.
- [100] Hauge S., Langseth H., Onshus T. (2006b). Reliability Data for Safety Instrumented Systems. PDS Method Handbook 2006 edition. SINTEF, Norway
- [101] Haupt R.L., Haupt S.E. (2004). Practical genetic algorithms. Wiley and Sons Inc. New Jersey, USA, 2004.
- [102] HID (2003). Guidance on As Low As Reasonably Practicable [ALARP] decisions in Control of Major Accident Hazards [COMAH]. Health and Safety Executive, Hazardous Installations Directorate, UK, Available from www.hse-databases.co.uk/hid/spc/perm12.htm. [last accessed 25-July-2003].

- [103]Hokstad P. (1988). A Shock Model for common-cause failures. *Reliability Engineering and System Safety* 1988;23(2):127-145.
- [104]Hokstad P. (2004). A generalisation of the Beta Factor Model. In Spitzer C., Schmocker U., Dang V.N. (eds.), *Procs. European Safety and Reliability Conference ESREL 2004 and International Conference on Probabilistic Safety Assessment and management PSAM7*. Berlin, Germany, 2004. p.1363-1368.
- [105]Hokstad P. (2005). Probability of Failure on demand (PFD) – the formulas of IEC 61508 with focus on the 1002D voting. In: Kolowrocki K. (ed.) *Proceedings of ESREL 2005*. Gdansk, Poland, pp 865-871
- [106]Hokstad P., Corneliussen K. (2004). Loss of safety assessment and the IEC 61508 standard. *Reliability Engineering and System Safety* 2004;83(1):111-120.
- [107]Hokstad P., Flotten P., Holmstrom S., McKenna F., Onshus T. (1995). A reliability model for optimization of test schemes for fire and gas detectors. *Reliability Engineering and System Safety* 1995;47(1):15-25.
- [108]Hokstad P., Maria A., Tomis P. (2006). Estimation of common cause factors from systems with different numbers of channels. *IEEE Transactions on Reliability* 2006;55(1):18-25.
- [109]Holland J. H. (1975). *Adaptation in Natural and Artificial Systems*. USA:University of Michigan Press.
- [110]Horn J., Nafpliotis N., Goldberg D.E. (1994). A niched Pareto genetic algorithm for multiobjective optimization. In: *Procs. first IEEE Conference on Evolutionary Computation*. IEEE World Congress on Computational Intelligence, vol 1, pp 82-87. Piscataway, New Jersey, USA.
- [111]HSE (2001). Reducing risk, protecting people. HSE's decision-making process. Health and Safety Executive, Her Majesty Stationery Office, Norwich, UK.
- [112]HSE (2002). Principles for Proof Testing of safety Instrumented Systems in the Chemical Industry. Contract Research Report 428/2002. Health and Safety Executive, Sheffield, UK, 2002.
- [113]Hussain A., Todinov M. (2007). Reliability optimization based on minimising the total system cost using genetic algorithms. In: Bartlett L. (ed.) *Procs. 17th Advances in Risk and Reliability Technology Symposium*, Loughborough, UK, 2007, p. 66-76.
- [114]Ida K., Gen M., Yokota T. (1994). System Reliability Optimization Problems with Several Failure Modes by Genetic Algorithm. In *Procs. 16th International Conference on Computers and Industrial Engineering*, Ashikaga, Japan, 1994. p.349-352.
- [115]IEC (1987). Technical Committee No 56: Reliability and Maintainability: Draft – Life cycle costing – Concepts, procedures and applications. Proposal for International standard for LCC. International Electrotechnical Commission, Switzerland.
- [116]IEC (1991). IEC 1078 Analysis techniques for dependability- Reliability block diagrams. International Electrotechnical Commission, Switzerland.
- [117]IEC (1995). IEC 1165 Application of Markov techniques. International Electrotechnical Commission, Switzerland.
- [118]IEC (1998-2005). IEC 61508 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. Parts 1-7. International Electrotechnical Commission, Switzerland..
- [119]IEC (2003). IEC 61511 Functional Safety – Safety Instrumented Systems for the Process Industry Sector. Parts 1-3. International Electrotechnical Commission, Switzerland.
- [120]ISA (1996). ISA S84.01-1996 Application of Safety Instrumented Systems for the Process Industry. The Instrumentation, Systems and Automation Society, USA.
- [121]ISA (1999). ISA TR84.0.02 Safety Instrumented Systems. Safety Integrity Level evaluation techniques. Draft Version 5. Parts 1-5. The Instrumentation, Systems and Automation Society, USA.
- [122]Jalote P. (1994). "Fault Tolerance in Distributed Systems". PTR Prentice Hall. New Jersey, USA.
- [123]Kawachi Y., Rausand M. (1999). Life cycle cost (LCC) analysis in oil and chemical process industries. Norwegian University of Science and Technology, Trondheim, Norway. Available at www.ntnu.no/ross/reports/lcc.pdf [last time accessed 09-July-2008].
- [124]Kim I.S., Martorell S., Vesely W.E., Samanta P.K. (1992). Quantitative evaluation of surveillance test intervals including test-caused risks. NUREG/CR-5775, BNLNUREG-52296. New York, US: Brookhaven National Laboratory, 1992.
- [125]Kim I.S., Martorell S., Vesely W.E., Samanta P.K. (1994). Risk analysis of surveillance requirements including their adverse effects. *Reliability Engineering and System Safety* 1994;45(3):225-234.
- [126]Knegtering B., Brombacher A. (1999). Application of micro Markov models for quantitative safety assessment to determine safety integrity levels as defined by IEC 61508 standard for functional safety. *Reliability Engineering and System Safety* 1999;66(2):171-175.
- [127]Knowles J.D., Corne D.W. (2000). Approximating the nondominated front using the Pareto archived evolutions strategy. *Evolutionary Computation* 2000;8(2):149-172.

- [128] Kohda T., Henley E.J., Inoue K. (1989). Finding modules in fault trees. *IEEE Transactions on Reliability* 1989;32(2):165-176.
- [129] Konak A., Coit D.W., Smith A.E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety* 2006;91(9):992-1007.
- [130] Kuo E., Rajendra-Prasad V., Tillman F.A., Hwang C.L. (2001). *Optimal reliability design. Fundamentals and applications*. Cambridge University Press, Cambridge UK.
- [131] Kuo W., Rajendra-Prasad V. (2000). An annotated overview of system-reliability optimization. *IEEE Transactions on Reliability* 2000;49(2):176-187.
- [132] Kuo W., Zuo M.J. (2003). *Optimal reliability modeling. Principles and applications*. John Wiley & Sons, New York, USA
- [133] Kvam P.H. (1998). A Parametric Mixture-Model for common-cause failure data. *IEEE Transactions on Reliability* 1998;47(1):30-34.
- [134] Kvam P.H., Martz F.M. (1995). Bayesian inference in a discrete shock model using confounded common cause data. *Reliability Engineering and System Safety* 1995;48(1):19-25.
- [135] Kvam P.H., Miller F.M. (2002). Common cause failure prediction using data mapping. *Reliability Engineering and System Safety* 2002;76(3):273-278.
- [136] Lapa C.M.F., Pereira C.M.N.A., Frutuoso e Melo P.F. (2003). Surveillance test policy optimization through genetic algorithms using non-periodic intervention frequencies and considering seasonal constraints. *Reliability Engineering and System Safety* 2003;81(1):103-109.
- [137] Laprie J.C. (1985). Dependable computing and fault tolerance: Concepts and terminology. In *FTCS-25, 25th International Symposium on Fault Tolerant Computing*. Pasadena, California, USA June 2005, p. 2-11.
- [138] Laprie J.C, ed. (1992). *Dependability: Basic concepts and terminology*. Springer-Verlag, Vienna, Austria.
- [139] Lee J.H., Chang S.H., Yoon W.H., Hong S.Y. (1990). Optimal test interval modeling of the nuclear safety systems using the inherent unavailability and human error. *Nuclear Engineering and Design* 1990;122(1-3):339-348.
- [140] Lees F.P. (1996). *Loss prevention in the process industry*. Volume 1. 2nd edition. Butterworth-Heinemann, Oxford, UK.
- [141] Leinum T. (1992). Reliability of heat detector systems on an offshore production plant. *Reliability Engineering and System Safety*, 1992, 36(1), 63-66.
- [142] Leveson N.G. (1995). *Safeware. System safety and computers*. Addison-Wesley Publishing Company, New York, USA.
- [143] Levitin G., Lisnianski A., Ben-Haim H., Elmakis D. (1998). Redundancy Optimization for Series-Parallel Multistate Systems. *IEEE Transactions on Reliability* 1998;47(2):65-172.
- [144] Lewis E.E. (1996) *Introduction to reliability engineering*. 2nd ed. John Wiley & Sons, New York, USA.
- [145] Li F., Pilgrim J.D., Dabeedin C., Chebbo A., Aggarwal R.K. (2005). Genetic algorithms for optimal reactive power compensation on the National Grid System. *IEEE Transactions on Power Systems* 2005;20(1):493-496.
- [146] Lisnianski A., Levitin G. (2003). *Multi-state system reliability. Assessment, optimization and applications*. World Scientific Publishing, Singapore.
- [147] Littlewood B. (1996). The impact of diversity upon common mode failures. *Reliability Engineering and System Safety* 1996;51(1):101-113.
- [148] Littlewood B., Miller D.R. (1989). Conceptual modelling of coincident failures in multi-version software. *IEEE Transactions on Software Engineering* 1989;15(12):1596-1614.
- [149] Lu D., Lutz R.R. (2002). Fault Contributions Trees for Product Families. *Proceedings of the 13th International Symposium on Software Reliability Engineering*, Annapolis, MD, USA, November 2002, p. 231-242.
- [150] Lu L., Jiang J. (2007). Analysis of on-line maintenance strategies for *k*-out-of-*n* standby safety systems. *Reliability Engineering and System Safety* 2007;92(2):144-155.
- [151] Lu L., Lewis G. (2006). Reliability evaluation of standby safety systems due to independent and common cause failures. In: *Proc IEEE International Conference on Automation Science and Engineering 2006*, Shanghai, China, 7-10 October.
- [152] Lu L., Lewis G. (2008). Configuration determination for *k*-out-of-*n* partially redundant systems. *Reliability Engineering and System Safety* 2008;93(11):1594-1604.
- [153] Lundteigen M.A. & Rausand M. (2008a). Architectural constraints in IEC 61508: Do they have the intended effect? *Reliability Engineering and System Safety* 2008, article in press.

- [154]Lundteigen, M.A. and Rausand M. (2008b). Spurious activation of safety instrumented systems in the oil and gas industry: Basic concepts and formulas. *Reliability Engineering and System Safety* 2008;93(8):1208-1217.
- [155]Lyndersen S., Aaro R. (1989). Life cycle cost prediction handbook; computer-based process safety systems. SINTEF report STF75 A89024. Trondheim, Norway.
- [156]Marseguerra M., Zio E. (2000). Optimizing maintenance and repair policies via a combination of genetic algorithms and Monte Carlo simulation. *Reliability Engineering and System Safety* 2000;68(1):69-83.
- [157]Marseguerra M., Zio E., Martorell S. (2006). Basics of genetic algorithms optimization for RAMS applications. *Reliability Engineering and System Safety* 2006;91(9):977-991.
- [158]Marseguerra M., Zio E., Podofillini L. (2004a). Optimal Reliability/Availability of Uncertain Systems via Multi-Objective Genetic Algorithms. *IEEE Transactions on Reliability* 2004;53(3):524-434.
- [159]Marseguerra M., Zio E., Podofillini L. (2004b). A Multiobjective Genetic Algorithm Approach to the Optimization of the Technical Specifications of a Nuclear Safety System. *Reliability Engineering and System Safety* 2004;84(1):87-99.
- [160]Martorell S., Carlos S., Sanchez A., Serradell V. (2000). Constrained optimization of test intervals using a steady-state genetic algorithm. *Reliability Engineering and System Safety* 2000;67(3):215-232.
- [161]Martorell S., Carlos S., Sanchez A., Villanueva J.F. (2007). Genetic Algorithms applications in surveillance and maintenance optimization. In: Leviti G (ed.) *Computational intelligence in reliability engineering*. Springer-Verlag, Berlin, 2007. Berlin, Germany.
- [162]Martorell S., Carlos S., Villanueva J.F., Sanchez A.I., Cepin M. (2005b). Optimization of surveillance requirements at NPP considering time-dependent scheduling of tests and preventive maintenance. In Kolowroki K. (ed), *Procs. European Safety and Reliability Conference ESREL '05*. Tri City, Poland, 2005. p.1363-1370.
- [163]Martorell S., Carlos S., Villanueva J.F., Sanchez A.I., Galvan B., Salazar D., Cepin M. (2006). Use of multiple objective evolutionary algorithms in optimizing surveillance requirements. *Reliability Engineering and System Safety* 2006; 91(9):1027-1038.
- [164]Martorell S., Sánchez A., Carlos S., Serradell V. (2004). Alternatives and Challenges in Optimizing Industrial Safety Using Genetic Algorithms. *Reliability Engineering and System Safety* 2004;86(1):25-34.
- [165]Martorell S., Sanchez A., Carlos S., Serradell V. (2002). Simultaneous and multi-criteria optimization of TS requirements and maintenance at NPPs. *Annals Nuclear Energy* 2002;29(2):147-168.
- [166]Martorell S., Sanchez A., Villanueva J.F., Carlos S., Serradell V. (2008a). A multi-objective genetic algorithm for RAMS+C optimization with uncertain decision variables. *Procs. of IMechE Part O, Journal of Risk and Reliability* 2008;222(O2):153-160.
- [167]Martorell S., Serradell V., Samanta P.K. (1995). Improving allowed outage time and surveillance test interval requirements: a study of their interactions using probabilistic methods. *Reliability Engineering and System Safety* 1995;47(2): 119-129.
- [168]Martorell S., Villanueva J.F., Carlos S., Nebot Y., Sánchez A., Pitchard J.L., Serradell V. (2005a). RAMS+C Informed Decision-Making with Application to Multi-objective Optimization of Technical Specifications and Maintenance Using Genetic Algorithms. *Reliability Engineering and System Safety* 2005;87(1):65-75.
- [169]Martorell S., Villanueva J.F., Carlos S., Sanchez A. (2008b). Maintenance modelling and optimization applied to safety-related equipment at nuclear power plants. In SAFERELNET project. Polytechnic University of Valencia, Spain. 2008. In preparation.
- [170]Martorell S.A., Melia M.E., Marti S.M., Garcia J.S., Soriano-Melchor F., Verdu-Martin G., Serradell V. (1988). Análisis del periodo de pruebas optimo en la revisión de especificaciones técnicas de C.N. relativas a STT's. Aplicación a un sistema por dos componentes en paralelo. XIV Annual Meeting of the Spanish Nuclear Society. Marbella, Spain, 1988.
- [171]McWilliams T.P., Martz H.F. (1980). Human error considerations in determining the optimum test interval for periodically inspected standby systems. *IEEE Transactions on Reliability* 1980;R-29(4):305-310.
- [172]Medoff, M. (2007). IEC 61508 Functional Safety Assessment. Project: 3144P Safety Certified Temperature Transmitter. Exida.com L.L.C. Sellerville, USA. Available online at: <http://www.exida.com/applications/sael/pdf/emerson/Emerson%2006-09-33%20R001%20V1R1%20%20IEC%2061508%20Assessment.pdf> [last time accessed March 2007].

- [173] Meulen v.d.M. (2003). On the use of smart sensors, common cause failure and the need for diversity. In: *Proc 6th International Symposium of Programmable Electronic Systems in Safety Related Applications*. TUV. Available on-line at www.sipi61580.com/ciks/vandermeulenm1.pdf. [last accessed 12-October-2008].
- [174] Mosleh A. (1991). Common cause failures: An analysis methodology and examples. *Reliability Engineering and System Safety* 1991;34 (3):249-292.
- [175] Mosleh A., Fleming K.N., Parry G.W., Paula H.M., Worledge D.H., Rasmuson D.M. (1988). *Procedures for Treating Common Cause Failures in Safety and Reliability Studies*. Vol. 1: Procedural framework and examples. Vol. 2: Analytical background and techniques. NUREG/CR-4780 (EPRI NP-5613). U.S.A: Nuclear Regulatory Commission; 1988.
- [176] Moustafa M.S. (1996). Transient analysis of reliability with and without repair for K-out-of-N:G systems with two failure modes. *Reliability Engineering and System Safety* 1996;53(1):31-35.
- [177] Moustafa M.S. (1997). Reliability analysis of K-out-of-N:G systems with dependent failures and imperfect coverage. *Reliability Engineering and System Safety* 1997;58(1):15-17.
- [178] Mulenbein H., Schlierkamp-Voosen (1993). Predictive models for the Breeder Genetic Algorithm. *Evolutionary Computation* 1993;1(1):25-49.
- [179] Muñoz A., Martorell S., Serradell V. (1997). Genetic algorithms in optimizing surveillance and maintenance of components. *Reliability Engineering and System Safety* 1997;57(2):107-120.
- [180] NORSOK (1996). O-CR-001 Life cycle cost for systems and equipment. *The Competitive Standing of the Norwegian Offshore Sector*, Norway.
- [181] Painton L., Campbell J. (1995). Genetic Algorithms in Optimization of System Reliability. *IEEE Transactions on Reliability* 1995;44(2):172-178.
- [182] Pattison R.L., Andrews J.D. (1999). Genetic algorithms in optimal safety system design. *Proc of IMechE Part E, Journal of Process Mechanical Engineering* 1999;213(E3):187-97.
- [183] Pham H. (1992). On the optimal design of k-out-of-n:G subsystems. *IEEE Transactions on Reliability*, 1992, 41(4), 572-575.
- [184] Pham H. (1993). Optimal cost-effective design of triple-modular-redundancy-with-spare systems. *IEEE Transactions on Reliability* 1993;42(3):369-374.
- [185] Pham H., Galyean W.J. (1992). Reliability analysis of nuclear fail-safe redundancy. *Reliability Eng. and System Safety* 1992;37(2):109-112.
- [186] Podofilini L., Zio E. (2008). Designing a risk-informed balanced system by genetic algorithms: Comparison of different balancing criteria. *Reliability Engineering and System Safety* 2008;93(12):1842-1858.
- [187] Purshouse R.C. (2003). On the evolutionary optimisation of many objectives. PhD Thesis. The University of Sheffield. Sheffield, UK.
- [188] Rachmawati L., Srinivasan D. (2006). Preference Incorporation in Multi-objective Evolutionary Algorithms: A Survey. In *Proc. IEEE Congress on Evolutionary Computation CEC 2006*, Vancouver, Canada, pp. 3385--3391.
- [189] Ramirez-Marquez J.E., Coit D.W. (2007). Optimization of system reliability in the presence of common cause failures. *Reliability Engineering and System Safety* 2007;92(10):1421-1434.
- [190] Rao K.D., Gopika V., Kushwaha H.S., Verma A.K., Srividya A. (2007) Test interval optimization of safety systems of nuclear power plant using fuzzy-genetic approach. *Reliability Engineering and System Safety* 2007;92(7):895-901.
- [191] Rausand M., Hoyland A. (2004). *System Reliability Theory. Models, Statistical methods and Applications*. 2nd ed. Wiley Interscience. New Jersey, USA.
- [192] Rechenberg I. (1973). *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, Stuttgart, Deutschland.
- [193] Reeves C.N., Rowe J.E. (2002). *Genetic Algorithms: Principles and perspectives. A guide to Genetic Algorithms theory*. Kluwer Academic Publishers. Norwell, MA, USA.
- [194] Riauke J., Bartlet L. (2008). An offshore safety system optimization using an SPEA2-based approach. *Proc. of IMechE Part O, Journal of Risk and Reliability* 2008;222(O3):271-282.
- [195] Rouvroye J.L., van den Blik E.G. (2002). Comparing safety analysis techniques. *Reliability Engineering and System Safety* 2002;75(3):289-294.
- [196] Rouvroye J., Brombacher A. (1999). New quantitative safety standards; different techniques, different results? *Reliability Engineering and System Safety* 1999;66(2):121-125.
- [197] Salazar D., Rocco C.M., Galvan B.J. (2006). Optimization of constrained multiple-objective reliability problems using evolutionary algorithms. *Reliability Engineering and System Safety* 2006;91(9):1057-1070.

- [198] Salazar D.E., Claudio M., Rocco S. (2007). Solving advanced multi-objective robust designs by means of multiple objective evolutionary algorithms (MOEA): A reliability application. *Reliability Engineering and System Safety* 2007;92(6):697-706.
- [199] Samanta P., Kim I.S., Mankamo T., Vesely W.E. (1994). *Handbook of Methods for Risk-Based Analyses of Technical Specifications*, NUREG/CR-6141. USA, Nuclear Regulatory Commission; 1994.
- [200] Schaffer J.D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In: Grefenstette JJ (ed.). *Proceedings of the First International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates, New Jersey, pp 93-100.
- [201] Schneeweiss W.G. (2001). Tutorial: Petri nets as a graphical description medium for many reliability scenarios. *IEEE Transaction on reliability* 2001;50(2):159-164.
- [202] Schofield S.L. (1993). Optimisation of proof test intervals in fault tree analysis. *Reliability Engineering and System Safety* 1993;41(2):127-133.
- [203] Schwefel H.P. (2000). Advantages (and disadvantages) of evolutionary computation over other approaches. In: Back T, Fogel DB, Michalewicz Z (eds.). *Evolutionary computation 1. Basic algorithms and operators*. Institute of Physics Publishing. Bristol, UK.
- [204] Shooman M.L. (1968). *Probabilistic reliability: An engineering Approach*. McGraw-Hill Book Company, New York, USA.
- [205] Shooman M.L. (2002). *Reliability of Computer Systems and Networks*. Wiley Interscience, New York, USA.
- [206] Signoret J-P. (1997). High Integrity Protection Systems (HIPS) – Making SIL calculations effective. *Exploration and production – oil and gas review (OTC edition)*, p. 14-17.
- [207] Signoret J-P., Duduit Y., Rauzy A. (2007). High Integrity Protection Systems (HIPS): Methods and tools for efficient Safety Integrity Levels (SIL) analysis and calculations. In Aven T. & Vinnem J.E. (eds.): *Procs. European Safety and Reliability Conference ESREL 2007, Stavanger, Norway, 25-27 June 2007*. London: Taylor & Francis.
- [208] Smith A.E., Tate D.M. (1993). Genetic optimization using a penalty function. In: Forrest S (ed.). *Procs. of the Fifth Conference on Genetic Algorithms, 1993*. Morgan Kaufmann Publishers, San Francisco, CA, USA, p. 499-505.
- [209] Smith D.J. (2005). *Reliability, maintainability and risk*. 7th edition. Elsevier Butterworth-Heinemann, Massachusetts, USA.
- [210] Srinivas N., Deb K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation* 1994;2(3):221-248.
- [211] Storey N. (1996). *Safety-Critical Computer Systems*. Addison Wesley Longman. New York, USA.
- [212] Suich R.C., Patterson, R.L. (1991). K-out-of-n:G systems: Some cost considerations. *IEEE Transactions on Reliability*, 1991, 40(3), 259-264.
- [213] Taboada H.A., Baheranwala F., Coit D.W., Wattanapongsakorn N. (2007). Practical solutions for multi-objective optimization: An application to system reliability design problems. *Reliability Engineering and System Safety* 2007;92(3):314-322.
- [214] Tao H., Liao G., Wang L. (2003). Integer coded genetic algorithm design of staggered sampling MTI. In: *Proc. IEEE International Conference in Neural Networks and Signal processing 2003*. Nanning, China. Vol 1, p. 558-562.
- [215] Tavakkoli-Moghaddam R., Safari J., Sassani F. (2008). Reliability optimization of series-parallel systems with a choice of redundancy strategies using a genetic algorithm. *Reliability Engineering and System Safety* 2008;93(4):550-556.
- [216] The Norwegian Industry Association (2004). *OLF-070 Application of IEC 61508 and IEC 61511 in the Norwegian Petroleum Industry*. Norway, 2004.
- [217] Tian Z., Zuo M.J. (2006). Redundancy Allocation for Multi-state Systems using Physical Programming and Genetic Algorithms. *Reliability Engineering and System Safety* 2006;91(9):1049-1056.
- [218] Uryas'ev S., Vallergera H. (1993). Optimization of test strategies: a general approach. *Reliability Engineering and System Safety* 1993;41(2):155-165.
- [219] Van Veldhuizen D.A. (1999). *Multiobjective Evolutionary algorithms: Classifications, analyses and new innovations*. PhD Thesis. Air Force Institute of Technology. Ohio, USA.
- [220] Vaurio J.K. (1994). The theory and quantification of common cause shock events for redundant standby systems. *Reliability Engineering and System Safety* 1994;43(3):289-305.
- [221] Vaurio J.K. (1995a). Optimization of test and maintenance intervals based on risk and cost. *Reliability Engineering and System Safety* 1995;49(1):23-26.
- [222] Vaurio J.K. (1995b). The probabilistic modelling of external common cause failure shocks in redundant systems. *Reliability Engineering and System Safety* 1995;50(1):97-107.

- [223] Vaurio J.K. (1998). An Implicit Method for incorporating common-cause failures in system analysis. *IEEE Transactions on Reliability* 1998;47(2):173-180.
- [224] Vaurio J.K. (1999). Common-cause failure models, data, quantification. *IEEE Transactions on Reliability* 1999;48(3):213-214.
- [225] Vaurio J.K. (2002). Treatment of general dependencies in system fault-tree and risk analysis. *IEEE Transactions on Reliability* 2002;51(3):278-287.
- [226] Vaurio J.K. (2003). Common cause failure probabilities in standby safety system fault tree analysis with testing-scheme and timing dependencies. *Reliability Engineering and System Safety* 2003;79(1):43-57.
- [227] Vaurio J.K. (2007). Consistent mapping of common cause failure rates and alpha factors. *Reliability Engineering and System Safety* 2007;92(5):628-645.
- [228] Vesely W.E. (1999). Principles of resource-effectiveness and regulatory effectiveness for risk-informed applications: Reducing burdens by improving effectiveness. *Reliability Engineering and System Safety* 1999;63(3):283-292.
- [229] Vesely W.E., Apostolakis G.E. (1999). Editorial. Developments in risk-informed decision-making for nuclear power plants. *Reliability Engineering and System Safety* 1999;63(3):223-224.
- [230] Vesely W.E., Goldberg F.F., Roberts N.H., Haasl D.F. (1981). *NUREG-0492 Fault Tree Handbook*. Nuclear Regulatory Commission. Washington D.C. USA.
- [231] Villanueva J.F., Sanchez A.I., Carlos S., Martorell S. (2008). Genetic algorithm-based optimization of testing and maintenance under uncertain unavailability and cost estimation: A survey of strategies for harmonizing evolution and accuracy. *Reliability Engineering and System Safety* 2008;93(12):1830-1841.
- [232] Vinod G., Kushwaha H.S., Verma A.K., Srividya A. (2004). Optimization of ISI interval using genetic algorithms for risk informed in-service inspection. *Reliability Engineering and System Safety* 2004;86(3):307-316.
- [233] Weile D.S., Michielsen E. (1996). Integer coded Pareto genetic algorithm design of constrained antenna arrays. *Electronic Letters* 1996;32(1):1744-1745.
- [234] Wilton S.R. (1998). Safety system logic solver availability, 1oo2D and TMR. *ISA Transactions*, 1998, 37(4), 353-358.
- [235] Xie L., Zhou J., Wang X. (2005). Data mapping and the prediction of common cause failure probability. *IEEE Transactions on Reliability* 2005;54(2):291-296.
- [236] Yang J., Hwang M., Sung T., Jin Y. (1999). Application of Genetic Algorithm for Reliability Allocation in Nuclear Power Plants. *Reliability Engineering and System Safety* 1999;65(3):229-238.
- [237] Yoshimura I., Sat Y. (2008). Safety achieved by the Safe Failure Fraction (SFF) in IEC 61580. *IEEE Transactions on Reliability* 2008;57(4):662-669.
- [238] Yu H., Chu C., Chatelet E., Yalaoui F. (2007). Reliability optimization of a redundant system with failure dependencies. *Reliability Engineering and System Safety* 2007;92(12):1627-1634.
- [239] Zhang T., Long W., Sato Y. (2003). Availability of systems with self-diagnostics components – Applying Markov model to IEC 61508-6. *Reliability Engineering and System Safety* 2003;80(2):133-141.
- [240] Zhao J., Chan A.H.C., Roberts C., Madelin K.B. (2007). Reliability evaluation and optimization of imperfect inspections for a component with multi-defects. *Reliability Engineering and System Safety* 2007;92(1):65-73.
- [241] Zio E., Di Maio F., Martorell S. (2008). Fusion of artificial neural networks and genetic algorithms for multi-objective system reliability design optimization. *Procs. of IMechE Part O, Journal of Risk and Reliability* 2008;222(O2):115-126.
- [242] Zio E., Podofillini L. (2007a). Integrated optimization of system design and spare parts allocation by means of multiobjective genetic algorithms and Monte Carlo simulation. *Procs. of IMechE Part O, Journal of Risk and Reliability* 2007;221(O1):67-84.
- [243] Zio E., Podofillini L. (2007b). Importance measures and genetic algorithms for designing a risk-informed optimally balanced system. *Reliability Engineering and System Safety* 2007;92(10):1435-1447.
- [244] Zitzler E., Deb K., Thiele L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation* 2000;8(2):173 – 195.
- [245] Zitzler E., Laumanns M., Thiele L. (2002). SPEA2: Improving the strength Pareto evolutionary algorithm. In: Giannakoglou K. et al. (ed.) *Procs. EUROGEN 2001, Evolutionary methods for design, optimization and control with applications in industrial problems*, pp 95-100. Athens, Greece.
- [246] Zitzler E., Thiele L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation* 1999;3(4):257-271.

APPENDIX A

Extended topics in reliability and safety

A.1. DETAILED REVIEW OF DEPENDABILITY MODELLING

A.1.1. Modelling methods

The main methods used for dependability measures in safety analysis are:

1. Simplified equations (SE). These are a set of equations, obtained from other methods, for specific architectures, and simplified for combination and merged for solution of larger systems (IEC 61508; ISA, 1999; Hauge et al., 2006a)
2. Reliability Block Diagrams (RBD). Representation of the system structure by functional blocks, showing graphically the conditions for successful operation of the system. (IEC, 1991; Rouvroye & van den Bliet, 2002).
3. Fault Tree Analysis (FTA). A graphical top-down representation of the different combinations between basic events (e.g. faults) leading to system failure; i.e. top event (Vesely et al., 1981)
4. Markov Analysis (MA). A method representing the different possible states of the system components, and the transitions among these states (IEC, 1995)
5. Petri Nets (Duduit et al., 1997; Duduit et al., 2008). Directed graphs composed of two types of nodes: states and transitions (Schneeweiss, 2001). Tokens are used for indicating the actual active states, and are moved from state to state to simulate transitions.
6. Hybrid methods. Combination of several techniques, like RBD, FTA and MA for solution of complex systems. (Knetering & Brombacher, 1999; Signoret et al., 2007; Duduit et al., 2008).

ISA TR84.0.02 (1999) specifically treats the application of SE, FTA and MA to Safety Instrumented Systems and makes a comparison of their modelling capability. SE is only able to handle simple systems. FTA (solved by Boolean algebra) can be used for modelling systems with more complex relationships. It can handle diverse repair times and diverse redundancy. Markov analysis (solved by matrix algebra) can do this as well, and in addition is capable of modelling time dependent requirements and handling sequence dependent failures. A big advantage of FTA over MA is that the former provides graphics that allow easy visualization of the failure paths.

Several authors have presented significant descriptions and comparative analysis of the different modelling techniques. Goble (1998) uses FTA and MA techniques for modelling several Moon architectures, including details such as CCF quantification and diagnostic coverage. These are validated and it is concluded that they provide similar results, however the MA approach is advantageous for including time dependency and the interaction of multiple failure modes.

Goble & Cheddie (2005) examined RBD and FTA. Both techniques provide a graphic representation of probability combinations. The main difference is that RBD focus on system success, while FTA focuses on system failure. Since FTA shows clearly the propagation mechanism of multiple failure modes, Goble & Cheddie express their preference for this in modelling SIS.

Rouvroye & Brombacher (1999) gave an account of FTA, RBD, MA and hybrid methods and contrasted them all exposing their advantages and disadvantages. As hybrid techniques they included the method proposed by IEC 61508-6 and the first edition of the PDS Method (Aaro et al., 1989). They show that RBD is the least comprehensive method, giving very pessimistic results. FTA and the two hybrid methods have approximately the same capabilities (inclusion of CCF, effect of diagnostics, and effects of test and repair, the latter only time-averaged) but FTA cannot include systematic failures (they do not indicate it, but the IEC method cannot either) and give practically the same results. For the authors Markov Analysis is the best method, although they propose to use a new method (hybrid in the author's view) called Enhanced Markov Analysis (EMA), which is a combination of MA, uncertainty analysis (via Monte Carlo simulation) and sensitivity analysis. In their results, they obtain a lower PFD_{avg} figure. Nevertheless, they included the probability of the system being in a safe state (i.e. down), which is considered by the author is not adequate for PFD_{avg} calculation. The analysis is built upon by Rouvroye & van den Blik (2002), reinforcing these conclusions.

Andrews & Ericson (2000) compared FTA and MA for several design complexities (e.g. partial coverage, standby systems). They concluded that FTA delivers either the same results or very good approximations to MA results for several cases. Although MA is more exact, it is frequently necessary to exclude many contributing events in order to simplify the model, which converts it into an approximation. To create Markov models for systems that are not very simple is difficult and error prone, and for big systems solutions can be only obtained by using numerical methods. FTA in contrast, is powerful for modelling large and complex systems easily, and their results are very good when small probabilities are involved (as is usual in safety systems).

Bukowski (2005) recently made a comparative analysis of MA and SE for quantification of PFD_{avg} . She points out that there is a vivid debate amongst advocates of MA, FTA and SE, some even arguing that MA models are incorrect. She concludes that SE are much easier to apply, but for getting rough estimates based on significant implications that may lead to significant errors, specially with undetected failures. On the contrary, MA when constructed and interpreted correctly gives the same results as those obtained by classical probability techniques. They do, however, require more expert understanding for their application.

IEC 61508-6 suggests a PFD_{avg} quantification method based on simplified equations (obtained from RBD). It presents the disadvantages of SE mentioned by the other authors above, since it seems to be oversimplified and not adequate for a detailed analysis of systems. Hauge et al. (2006a) in the PDS Method presents a more refined technique based on calculation formulas, which is presented with an example with simple RBD. The method aims to include failure categories and causes excluded by previous techniques. It presents a very novel approach to quantification of CCF by the β factor model, which is incorporated in this thesis. More recently, Guo & Yang (2007) proposed an RBD based approach, which has an equivalent mathematical characteristic to FTA, and addresses and improves the approach originally proposed by IEC 61508 Part 6 based on simplified equations derived through RBD for SIL verification.

A.1.2. Modelling PFD for SIL analysis

IEC 61508 Part 6 provides a suggested methodology for quantification of PFD_{avg} detailed in Section 1.6.2. As mentioned before, it is based on simplified equations derived from RBDs. It also provides tables for the PFD_{avg} of a few specific architectures for fixed combinations of failure rate, diagnostic coverage and β factor.

Goble (1998) provides an alternative to modelling PFD_{avg} . As mentioned above, it uses either FTA or MA. The reasons for considering FTA more advantageous have been outlined above. Goble provided the detailed modelling of several architectures up to three components, including modelling detail such as safe and dangerous failures, CCF and diagnostic coverage. This is the base used in this thesis for the fault tree modelling. Goble bases his modelling on using the failure modes division as presented in Eq. (1.20). He solved the FTs by deriving simplified equations and obtaining the average value by integration.

The Markov modelling is based on previous work with Bukowski (Bukowski & Goble 1995). Bukowski and Goble (2001) and Bukowski (2001) have continued developing the MA method for safety systems. Zhang et al. (2003) presented a new analysis of MA for systems with self-diagnostic components.

The methods presented by ISA TR84.0.02 (ISA, 1999) include SE, FTA and MA. They use the same failure modes taxonomy as Goble (1998), which is also used in this thesis (Eq. (1.20)). The standard includes terms for quantification of systematic failures. It proposes for FTA the addition of basic events for systematic failures in the same fashion as for CCF. However, to obtain statistical data for systematic failure probability or rates is very difficult. The standard later drops the term for systematic failure rate when presenting the set of equations with further simplifications. The standard itself recognizes that it is very difficult to quantify systematic failures contribution due to the diversity of causes of failures (specification, design, implementation, operation, etc.). Goble (1998) and ISA TR84.0.02 Part 3 are the two main references for FTA applied to safety systems with two failure modes and with sufficient detail for application to IEC 61508.

Knegtering & Brombacher (1999), Rouvroye & Brombacher (1999) and Rouvroye and van den Blik (2002) proposed the use of hybrid Markov methods for solution of PFD_{avg} : Micro Markov models, and Enhanced Markov Analysis. The former implies to introduce small Markov models for sub-divisions of RBDs, which are easier to solve and reduces the necessary operations to execute for quantifying the PFD_{avg} . It still, however, suffers from some degree of complexity compared to FTA, although it adds accuracy to RBD. The enhance Markov analysis is a combination of MA, uncertainty analysis (via Monte Carlo technique) and sensitivity analysis (Rouvroye & Brombacher, 1999). It seems to be a powerful technique, but with the same drawbacks of the basic MA plus the complexities of the additional techniques.

SINTEF proposed a new analytic method for quantification of reliability for process safety systems (Bodsberg & Hokstad, 1995, 1997). It is called the PDS Method (a Norwegian acronym for “reliability and availability of computer-based safety systems”). It started to explore the creation of an alternative failure taxonomy that relates directly failure cause, consequence and their means of improvement. This started addressing the need of using reliability calculations results for LCC quantification, in order to find cost-effective designs and operating philosophies. They proposed to use modified RBDs for development of the system logic model in order to get analytical expressions. At a later stage (Hokstad & Corneliusen, 2004), they started to identify the weaknesses in the IEC 61508 methods, proposing their new failure classification and suggesting an extended and more precise β factor model for quantification of CCF (further developed on Hokstad (2004) and Hokstad et al. (2006)). These developments gave as a result the final PDS Method, which is now at a second edition (Hauge et al., 2006a).

The PDS Method offers an alternative to quantification of PFD for compliance with IEC 61508. It aims to follow the IEC 61508 philosophy but with a more sophisticated and precise

dependability quantification methods. It intends to provide a methodology for realistic quantification of safety system performance in application-specific conditions (opposed to the laboratory-like environment). It considers three failure modes: dangerous, spurious trip and non-critical (those that do not affect the system functioning). The PDS method argues that although IEC 61508 excludes systematic failures from calculations, it implicitly includes a fraction of them in the random failure data, and in the quantification of CCF by the β factor, and a fraction of the diagnostic coverage.

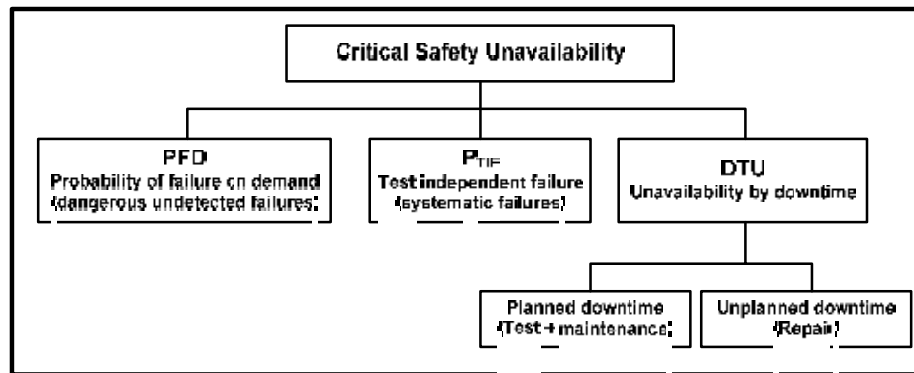


Figure A.1. Contributions to Critical Safety Unavailability in the PDS Method

The loss of safety measure proposed by SINTEF (Fig. A.9), called the Critical Safety Unavailability (CSU), includes:

- Unavailability due to dangerous undetected failures. This is quantified by the average PFD in this method.
- Systematic test-independent failures P_{TIF} , which are not detected either by the self-diagnostic mechanism nor the proof test. These are modelled as probabilities.
- Unavailability due to test and maintenance downtime DTU. This includes planned downtime (test and preventive maintenance) and unplanned (repair). These contributions depend on the operational philosophy. Something no other author had mentioned before.

The PDS method provides a set of simplified formulas for the factors mentioned above, and for quantification of STR as well.

PDS is largely focussed on modelling of MooN voting systems. PDS modifies the IEC 61508 β factor model by introducing a modification factor. This modifies the β in order to take into account the difference in CCF for different voting architectures. This is further discussed in Chapter 4. The PDS method provides a set of generic simplified equations that can be applied for any MooN voting configuration. They are three sets, one for each of the factors of CSU outlined above.

The PDS method is perhaps more comprehensive and generic than the one of IEC 61508-6. However, there exists the difficulty of having explicit input data for all the parameters. This is especially difficult for the systematic failure probability P_{TF} . The PDS method is classified as a Hybrid method (it uses simple RBDs), but is largely based on SE, and therefore it presents some of their drawbacks (oversimplification, difficulty to handle test and repair, etc.). However, its contribution is enormous, especially considering its provision for quantification of CCF and independent failure rates of different Moon architectures.

Guo & Yang (2007) intend to provide a detailed guidance for application of RBD, deriving simplified equations, compensating the lack of sufficient detail of IEC61508-6. Their method is based on the concept of mean down time, the same as IEC 61508, but they obtained a different expression. It is a great advance when compared with IEC 61508-6, but it still has some of the same drawbacks of the method provided by the standard.

Signoret et al. (2007) and Dudit et al. (2008) have proposed the use of fault trees for quantification of PFD_{avg} complemented with more powerful methods so that time-dependencies can be handled, and test and maintenance properly modelled. This constitutes hybrid methods, based on fault trees. The idea is to introduce multi-phase Markov models or Petri Nets substituting sub-modules of the fault tree, included as basic events. In this thesis, a similar idea has been developed (Chapters 5 and 6), but introducing time-dependent analytical models that comprises entire sub-systems.

A.1.3. Modelling of STR

Goble (1988) includes quantification of Probability of Failing Safely in the modelling with FTA and MA. It considers a sufficient level of detail to include diagnostic coverage and CCF. It presents the models for several architectures with up to three components. The measure actually is an unavailability by safe failure ($\lambda \cdot SD$, being SD the time to re-establish the process after a spurious shutdown). The contribution of Goble is significant in the sense that includes the analysis of safe failures giving them as much importance as dangerous failures, and analyzes them with the same level of detail. The measure for safe failure included by ISA TR 84.0.02 is the Mean Time to Spurious Trip ($MTTF^{spurious}=1/STR$). It actually quantifies the STR by SE, FTA and MA and then converts them to MMTF. In the quantification of STR by SE, it includes the dangerous detected failures. Albeit this may be correct, it depends on the operational philosophy of the system (Hauge et al., 2006a). Also in the SE method, it includes quantification of systematic failure rates. The same as for PFD quantification, Goble (1988) and ISA TR84.0.02 use the failure modes classification provided by Eq. (1.20). Notice, however,

that ISA TR84.0.02 drops one level of modelling detail (i.e. separation of detected and undetected failures) in the Markov examples, in order to avoid the growing complexity problem.

As mentioned above, the PDS Method (Hauge et al., 2006a) presents a quantification alternative to IEC 61580-6, still intending to comply with the requirements of the standard. The PDS Method acknowledges the fundamental importance that the quantification of consequences of safe failures have for SIS, classifying them as critical failures, given that they affect the basic/main functioning of the system. They conceptualize STR as a measure of the system ability to maintain production when safe. Therefore, STR is a measure of loss of production. Different from PFD_{avg} , the method does not include quantification of systematic failures for STR.

A set of generic equations for evaluation of STR for different architectures is given. The STR quantification method includes the same benefits as for the loss of safety quantification approach: The method is oriented to contemplate the different effects of the specific MoonN voting architecture, embedded in the modified β factor model.

There is a difference of opinion whether the dangerous detected failures should be treated as safe, and therefore should contribute to the STR. The PDS method recognizes that this is possible, but that it cannot be taken for granted, since it depends on the system operation philosophy. It is worth noting that the method does not include explicitly the dangerous detected failures in the STR quantification formulas.

The benefits and drawbacks of the PDS method for loss of production are the same as for loss of safety. Nevertheless, their contribution to the quantification of CCF with the modified β factor model is notable. The method also establishes firmly the necessity of quantifying STR in order to keep a balance between loss of safety and loss of production, and not only as a secondary performance metric.

Lu & Lewis (2006, 2008) presented analytical models for probability of spurious operation (and unavailability) based on the binomial distribution. They made a sensitivity analysis to changing k and n in k -out-of- n systems, and found graphically feasible regions of design according to specific requirements. Practical case studies for CANDU NPPs, which use 2oo3 and 2oo4 architectures were presented. This is useful for having an insight into the comparison of MoonN architectures. The study is extended in Lu & Jang (2007) to include the effects of on-line test and maintenance. The contribution of this work is important because it introduces the evaluation of the relative time the system spends in normal operation and test and maintenance, during

which its probability of spurious operation changes. This concept is considered in Chapter 6 herein. Apart from that, the level of modelling detail is too basic, not adequate for the study we intend to develop herein.

The first monothematic study about spurious activation of SIS was presented by Lundteigen & Rausand (2008b). This article defines and clarifies concepts related to spurious activation of SIS. It discusses the multiple causes of spurious trips and presents several analytical expressions for quantification of STR including the following factors: internal failures, dangerous detected failures, false demand, and loss of utilities. The first two are based on the binomial distribution, and they include CCF. The last two are important to quantify; however for comparison of competing architectures they are irrelevant. Notice that the authors clarify that the inclusion of dangerous detected failures as contributors to STR depends on the system configuration and operating philosophy. Systematic failure is acknowledged as a contributing factor, but it is not included in the quantification. They develop a set of simplified equations for a few architectures, and compare results against the equations given by the PDS method and ISA TR84.0.02. They conclude that the results are quite similar.

A.1.4. Fault Tree Analysis

A fault tree is a graphic analytical model that represents the interrelationship between a potential critical event and the combination of events that cause that event. The fault trees are constructed using symbols. The most important symbols are shown in Figure A.2. There are several other symbols of less common use. A full reference about the fault tree method and these symbols can be found in Vesely et al. (1981) and Andrews & Moss (2002). The application of FTA specific to SIS is well explained in Goble (1988) and ISA TR84.0.02. The steps for execution of a complete fault tree analysis (Leveson, 1995; Rausand & Hoyland, 2004) are given below:

1. Definition of the problem and boundaries. This requires to define clearly and concisely the critical event, which becomes the top event of the fault tree. For SIS this is usually system dangerous failure or system safe failure.

2. Fault tree construction. The fault tree is constructed from the top event down. The immediate causes of the top events are identified and connected to the top event using logic gates. The causes of these intermediate events are then identified and structured in the same way. This process is repeated until the root causes are found. The basic events can be for example faults or operator actions, and are delimited by the limit of resolution of the problem. It is fundamental to be able to include in the fault tree all the causes that are necessary and sufficient for the top event to occur.







	Top or intermediate event	Event that occurs as a combination of other events
	Basic event	Causal events of the lowest level in the fault tree
	AND gate	The output occurs if all inputs occurs simultaneously
	OR gate	The output occurs if any one of the inputs occurs
	Transfer symbol	Indicate a part of the tree that is developed somewhere else
	House event	Logic event that occurs as an external or trigger event, with true or false value

Figure A.2. Fault tree symbols

3. Qualitative analysis of the fault tree. The qualitative analysis of the fault tree intends to visualize paths of propagation of failure, and to identify the most critical events and weaknesses. When the fault tree is very small and simple this can be done just by direct analysis. Otherwise, the analysis requires reducing the tree to a *logically equivalent form, showing the specific combinations of basic events sufficient to cause the top event* (Leveson, 1995). This is the identification of minimal cut sets (CS). A CS is a set of basic events that combined ensure the top event occurs. When a cut set cannot be further reduced it is called a minimal cut set (MCS). The order of a cut set is the number of basic events that compose it. Identification of the MCS permits visualization of which combinations of basic events lead to the immediate occurrence of the top events. Therefore, low order cut sets, e.g. first and second order ones, are the most influential combinations in system failure and risks. MCS reduction for small fault trees can be practiced by hand, and computer algorithms are available for more complex situations (see Andrews & Moss, 2002)

4. Quantitative analysis of the fault tree. The quantitative analysis permits calculation of the likelihood of occurrence of the top event, based on the likelihood of the basic events. This can be expressed as a probability or frequency. The analysis is usually based on the MCS of the fault tree.

The basic gates of fault trees, AND and OR have a one-to-one correspondence to the Boolean

operations union and intersection (Andrews and Moss, 2002), and they can be solved using probability rules.

AND gates. The output of an AND gate is active if all its inputs occur simultaneously, being a Boolean intersection. If the events in the inputs are independent, the probability of two events occurring is determined by the multiplication probability law:

$$P(A \cap B) = P(A) \times P(B) \quad (\text{A.1})$$

Notice that the probability figures can be frequencies as well. However, it is a frequent mistake to multiply frequencies with frequencies, which gives meaningless results.

When the input events are not independent, the conditional probability law applies. Nevertheless, AND gates do not contemplate this case. In safety system analysis, the dependencies are quantified with other techniques.

OR gates. The output of an OR gate is active if any one or more of its inputs occur, being a Boolean union. Mutually exclusive events therefore are quantified by the addition probability law:

$$P(A \cup B) = P(A) + P(B) \quad (\text{A.2})$$

While non-mutually exclusive events are:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \quad (\text{A.3})$$

For more than two inputs:

$$P(A \cup B \cup C) = 1 - [(1 - P(A)) \cdot (1 - P(B)) \cdot (1 - P(C))] \quad (\text{A.4})$$

As Goble (1988) points out, in order to simplify and speed up calculations events of fault trees are frequently assumed to be independent or mutually exclusive events, so that the multiplication law or the addition law can be applied respectively. These assumptions provide valid answers when probabilities are low. In the case of mutually exclusive events in OR gates the approximate results are also in the conservative direction. In the case of AND gates, when there are significant dependencies that can affect the result (CCF), they are treated as separate basic events. When quantifying dependability for competing architectures using the same assumptions, these assumptions can be sufficient to get results to enable valid comparisons.

The top event probability can be quantified using several methods (Andrews & Moss, 2002):

- Gate-to-gate quantification. The basic event probabilities can be worked up through the tree until the top event probability is calculated. This is applicable when the fault tree is not too big, when the basic events are independent and there are not repeated events.

- Minimal cut sets. The MCS obtained in the qualitative analysis are used. This is applicable when there are repeated events in the fault tree, provided that the basic events are independent.
- Computer codes. When the fault trees are too big or too complex computer codes can be used. One very popular algorithm is called MOCUS, and several programs have been developed based on it. MOCUS is applicable when the fault tree does not have mutually exclusive events.

There are some other developments in fault tree analysis worthy of discussion. The most relevant to this thesis is the use of house events in the fault tree structure so that changing conditions can be handled (Andrews, 1994). The second most relevant is the modularization of fault trees (Chatterjee, 1975; Gulati & Dugan, 1997), which permits the solution of independent sub-structures of the tree by another method. This can be used for incorporating techniques that can handle time dependencies. These two techniques are discussed in Chapters 3 and 5 respectively.

Example: System dangerous failure. An example of a SIS is considered to illustrate the construction of fault trees. The fault tree is shown in Figure A.3, while the system under analysis is shown in the inset. The sensor subsystem has two redundant sensors. The PLC (logic solver) and the valve are non-redundant.

The fault tree defines system dangerous failure, which becomes the top event. Notice that analysis of system safe failure would require a separate fault tree. The failure modes classification includes diagnostic coverage and CCF (see Eq. (1.20)). The immediate causes of the top event are the dangerous failures of each subsystem: Sensors, PLC and valve. Thus, three intermediate events, one for each, are added under the top event. The dangerous failure of any subsystem would be sufficient to fail the entire system. Any of the three, either the failure of the sensor to detect a hazardous condition, the failure of the PLC to command an action to prevent its development, or the failure of the valve to execute the commanded action (close) would be sufficient to fail the entire system. Therefore, an OR gate is put in place. Observe now the immediate causes of the PLC's failure. Given by the action of the diagnostic coverage, there are two causes: a dangerous detected failure and a dangerous undetected failure. These are considered basic events because this a maximum level of resolution considered convenient for the analysis, and because this is the lowest level for which failure rate data is available in databases. Since any of the two failures would fail the PLC, an OR gate is used. It is assumed that the component can only fail with either a detected failure or an undetected failure, but not both; i.e. mutually exclusive events. In the same fashion, all eight failure modes in a component

(Eq. (1.20)) are considered to be mutually exclusive events. This is important to remember for the quantification of the tree. The valve also has only two basic events.

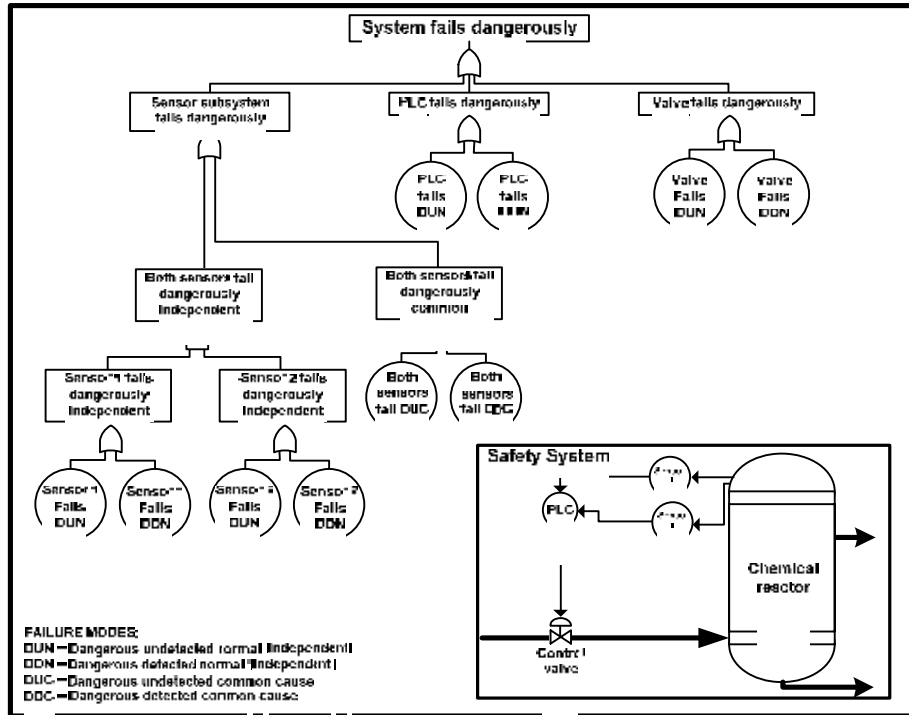


Figure A.3. Example of fault tree construction

The sensor subsystem modelling becomes more complex because it has two components, and therefore the phenomenon of CCF is present. Thus, the sensor subsystem can fail either by simultaneous independent failure of both sensors or by CCF. This means that the failure of both sensors is not independent of one another. The CCF dependency is handled as a separate intermediate event. This permits quantification of the AND gate using the basic multiplication law. Again, the action of diagnostic coverage creates two basic events for each sensor and for the CCF event.

A.2. CCF MODELLING

Common Cause Failure is the term used to define the event of the simultaneous failure of several components due to the same cause. There is some confusion between the terms CCF and Common Mode failure (CMF). Smith (2005) clarifies that CMF defines coincident failures of the same mode; i.e. they have the same appearance or effect. As it has been seen above, CCF is about simultaneous failures originated by the same cause. CMF are usually originated by a

single cause, although not necessarily. In this case they can be considered a subset of CCF. In order to avoid confusion, another term used for CCF is *dependent failures* (Lees, 1996). The name is self-explanatory.

Several different models have been proposed for modelling CCF. They are classified as parametric and non-parametric models. Parametric models intend to estimate the probability of CCF based on one or more parameters. Mosleh et al. (1988) presented an account of the main parametric methods used for quantification in safety and reliability studies. He categorized the parametric models as shock or non-shock models. Non-shock models include the best known models for safety and reliability studies: β Factor, Multiple Greek Letter and the Alpha Factor methods. For shock models, the binomial failure rate model is mentioned.

In brief, shock models take into account the frequency of shocks impacting the system and the conditional probability of failure subject to those shocks. On the contrary, non-shock models estimate directly the probability of CCF without attempting to quantify the influence of the shock events. The *Basic Parameter model* is intended to estimate directly the probability of CCF of basic events (for example per component). Given the difficulty to accomplish this, the other non-shock models rather seek to estimate parameters that permit determination of the ratio of multiple failures (CCF) to single failures (independent) of a component. The *β factor model* has been detailed in Section 4.1.2. The description given by Mosleh et al. (1998) about the other models is summarized as follows:

The *Multiple Greek Letter (MGL)* model is a multi-parameter model that assumes that the fraction of failures attributed to a common cause event differs dependent on the number of components failed. It therefore uses several parameters depending on the size of the group of components failed. In this fashion, β would correspond to the fraction of CCF for two components, γ for three components, δ for four components, and so on.

The *Alpha factor model* assumes also different parameters $\alpha_1, \alpha_2, \dots, \alpha_n$, depending on the number of components failed. The difference with the MGL model is that its parameters are estimated from component failures, while the alpha parameters are estimated from system failure data.

The *Binomial Failure Rate model* is a shock model that considers two types of shocks: lethal and non-lethal. Non-lethal shocks cause each component within the common cause component group to have a constant and independent probability of failure. The distribution of the number

of failed components is determined by a factor with a binomial distribution (hence the name of the model). Lethal shocks, in contrast, cause all components in the group to fail (with a conditional probability of unity). The complete model should include both non-lethal and lethal shocks. It therefore requires estimates of their frequency of occurrence and the conditional probability of failure of the components given non-lethal shocks.

Smith (2005) mentions some others models. One is the *Boundary model* (called the Geometric model in Lees (1996)), which uses the upper and lower bounds of failures of the system. The model assumes that the lower bound is related to the fact that all failures are independent (λ_i), and the upper bound that all failures are CCF (λ_u). Then, the total failure rate of the system is calculated by the equation:

$$\lambda^T = (\lambda_l^n \cdot \lambda_u)^{1/(n+1)} \quad (\text{A.5})$$

The parameter n is chosen according to the degree of diversity of the system, with a value between 1 and 4. Smith (2005) argues that the parameter n is selected in a very subjective way, with no sound relation between its value and the real causes of CCF.

There are several other developments in CCF modelling. Hokstad (1988) presented the *Random Probability Shock* (RPS) model, which models several different degrees of dependence amongst components as outcomes of shocks; i.e. the number of components that fail due to a shock. It requires estimating the shock rate and a dependence parameter. The parameter p (which determines the number of failed components) follows a beta-binomial distribution. Kvam (1998) formulated what he called a *Parametric Mixture* model, that is quite similar to the RPS model (as argued by Vaurio (1999)).

Recently, the technique of data mapping for estimating CCFs has been receiving significant attention. As Kvam & Miller (2002) indicate, this involves translating existing CCF data from one system with a specific size (number of components) into a system of different size. Earlier studies on mapping algorithms are presented in Mosleh et al. (1998) and Mosleh (1991). Recent developments include works by Kvam & Miller (2002), Xie et al. (2005) and Vaurio (2007).

Some guidance about how to estimate multiple parameters of the MGL model is for those provided by CCPS (2000) and for this and other models in Mosleh et al. (1998).

A seminal work presenting a complete methodology for treating CCF in safety and reliability studies is Mosleh et al. (1988). A summary was presented in Mosleh (1991). Mosleh et al.

(1988) identified the three essential factors to consider when analysing:

- The root cause of the failure, which is the event or factor that leads to CCF
- The couple mechanism.
- The lack of defences against CCF mechanisms.

The generic procedural framework for CCF analysis defined in Mosleh et al. (1988) comprises the following steps:

1. System logic model development.
2. Identification of common cause component groups.
3. Common cause modelling and data analysis.
 - Definition of common cause failure events.
 - Selection of probability models for common cause basic events.
 - Data classification and screening.
 - Parameter estimation (this is the step where CCF model is chosen and applied).
4. System quantification and interpretation of results.

A.3. REVIEW OF MooN ARCHITECTURES

A.3.1. Definition of MooN architectures

A MooN system was defined in Section 1.3 as a system with N units (i.e. components, channels, etc.) in which M out of them are sufficient to initiate the safety function. It requires a minimum of M units to “vote” for the execution of the safety function. It also implies that these M units must be functional for the system to be successful, and that a voting mechanism is provided. This is equivalent to a k -out-of- n :G with an added voter. There are, however, several diverse terms to designate the same or similar configurations, which can be confusing. A review of terms is given on the next.

Billington & Allan (1983) define as *partially redundant systems* those requiring some proportion of components between the two extreme of series (non-redundant) and parallel (fully-redundant) to operate. They mention *majority voting* and *m-out-of-n* systems as two other designations of this category. Although there are several designations for this type of systems, *k-out-of-n* is maybe the most used. Dhillon & Singh (1981) mention that these systems require a specified number of k units to be good for the system to succeed (in performing its intended function). To differentiate for subsystems that fail when k out of n units do (k -out-of- n :F), they

are called k -out-of- n :G. Notice that k -out-of- n are active (no standby) systems. Other terms used for the same description are m/N (Lewis, 1996) and r -out-of- n (Shooman, 1968, 2002).

Another term is N-Modular redundancy (NMR) (Storey, 1996), which is a static redundancy technique that employs fault masking (rather than fault detection). This is widely used for fault tolerant computer systems. In the framework of IEC 61508, this can be seen as a programmable electronic system used as a logic solver. Gopal et al. (1978) used the expression general modular redundant (GMR) subsystems for a group of statistically independent (s-independent) redundant subsystems, either in series or in a more complex structure. Biernat (1990, 1994) made a distinctive mention for k -out-of- n :G and NMR systems, where a NMR system is a particular case of a k -out-of- n :G system, and where $n=2k-1$ system (e.g. 2oo3). The most basic NMR structure is the Triple Modular redundant TMR (Shooman, 2002). In addition, Storey (1996) and Shooman (2002) consider that a NMR architecture includes the use of a majority voter together with the redundant structure. There has been mention of other hybrid cases such as Nuclear Fail Safe Redundancy NFSR (Pham & Galyean, 1992), a k -out-of- $(n+S)$, where S is a pool of spares and $k=n-1$, and Triple Modular Redundant with spares TMRWS. However, the NMR is the general and most approached case.

As it can be seen from the discussion above, NMR may be generally treated as an special case of k -out-of- n :G systems. For computer-based systems reliability this requires the addition of a voter. Remember also for the case of MoonD systems mentioned in Section 1.3 (voting architectures with added self-diagnostics, included in Gruhn & Cheddie (1998), Goble (1998) and Goble & Cheddie (2005)). These systems include the diagnostic circuit output as an extra input to the voting scheme (in substitution of a faulty unit). This would not apply to units that include automatic built-in diagnostics where the diagnostic circuit does not take any action upon failure detection other than only to announce the fault. In addition, these references include cases of redundancies where $M=\frac{1}{2}N$ (where N is an even integer). For instance, some safety PLCs present 2oo4 architectures (called quad systems). Therefore, the definition that NMR as a k -out-of- $(2k-1)$:G system does not perfectly match MoonN (and MoonND) systems used for SIS. This was explained by Shooman (2002), who mentions that, although usually an NMR system has an odd number for N , if additional information about a unit malfunction and a means to lock it out is available, it is feasible to have an even value for N . It can be inferred that this is the case for 2oo2D voting architectures in SIS. There are some more advanced complex techniques, a brief description of which is presented in Appendix B.

Shooman (2002) presents the example of a TMR, which is modelled as a 2-out-of-3 system (k -out-of- n , where $k=2$, $n=3$) based on the assumption that the (digital) elements fail such that they

produce the complement of the correct input (and that the units are independent and identical). Additionally, notice that a NMR system can have diverse degraded configurations (e.g. 3-2-0 or 3-2-1 for a TMR).

It can be deduced that a NMR system is basically a k -out-of- n system plus a voter. NMR can be seen as k -out-of- n :G system assuming that:

- The voter is perfect (reliability is equal to 1).
- The k represents the majority voting number for giving the correct voter output.
- The system units fail such that they produce the complement of the correct output.

Concluding, we refer to MooN systems to those which need at least M (out of the N) units to be good to perform its intended function successfully by M majority voting. Thus, MooN voting systems are k -out-of- n active standby systems (in series with a voter). When they are referred to as MooND (e.g. 1oo1D, 1oo2D) this will imply that the diagnostic circuit provides an additional input to the voter (although they are out of the scope of this thesis).

A.3.2. Advanced voting techniques in NMR systems

The advanced voting techniques (Shoorman, 2002) are basically techniques used in software fault tolerance (some based on McAllister & Vouk (1996)), but can be extrapolated to hardware or mixture hardware-software:

- **Adjudicator algorithms.** An adjudicator is the generic name for the voter element. An adjudicator algorithm for majority voting is a specific case where simply $n+1$ or more out of $N=2n+1$ must vote in agreement, where n is an integer and >0 and N is usually an odd number of elements. However, if N is to be either odd or even and an integer ≥ 3 , where m components must vote in agreement, this becomes an m -out-of- N system (see formula for m presented by McAllister & Vouk (1996) below).
- **Consensus voting.** This is an architecture where even the majority vote fails there can be an agreement among the rest of the components. This is of course more complex than majority voting. The value of k can even be equal to one, using a tie-breaking algorithm (e.g. some internal test). This seems to be the basis for systems different than 2oo3.
- **Test and switch techniques.** This tests the components, so that when one votes in disagreement the output of the system is decided with only the components whose test is valid (i.e. an acceptance test). This resembles the 1oo2D and 2oo2D architectures.

- **Pairwise comparison.** A comparison between outputs of modules $2n$ components, and where $N=2n$ (and $n>1$) determines the system output. This seems to be the logic behind 2oo4 systems.

In reality, McAllister & Vouk (1996) designate all the software voting techniques as “adjudication”. This can be done by majority voting or consensus voting.

- **Majority voting.** In an m -out-of- N tolerant system, where m is the agreement number, generally $m = \lceil (N+1)/2 \rceil$, where $\lceil \cdot \rceil$ represents the ceiling function. The particular case of 2-out-of- N is mentioned, where the agreement is always made by two components, assuming a large output space and statistical independence.
- **Consensus voting.** This a generalization of majority voting, where a more complex algorithm is used to select the system output if there is not an agreement with $m \geq \lceil (N+1)/2 \rceil$. The first option is to pick a unique agreement $< \lceil (N+1)/2 \rceil$. The second one is to employ a tie-breaking algorithm; specifically, to choose randomly or by acceptance test on component to decide the vote. It would be congruent to consider this technique as the basis of MooND systems.

A.3.3. Modelling of 1oo2D systems

The approach given by IEC 61508-6 has caused controversies in the quantification of dependability of 1oo2D architectures. This is due to the confusion of whether this architecture really behaves like a 2oo2D or a 1oo2D voting system (the definition provided by IEC 61508 seems to suggest that it actually behaves like a 2oo2D). Zhang et al. (2003) re-analyzed this case using MA, while Hokstad (2005) re-studied the case of 1oo2 and 2oo2 for comparison with 1oo2D, and redrew the formulas based on RBD. Guo & Yang (2007) reassessed the RBD method, with focus on clarification of the concept of MDT. The conclusion is that the best approach is to analyse every specific case, and not to simply stick to the IEC definition.

APPENDIX B

Advanced topics in Genetic Algorithms

B.1. GENETIC DRIFT AND NICHING

Genetic drift is the effect of the population evolving towards localized regions of the search space, promoting the convergence of the search to a small region of the trade-off surface, which reduces the efficiency of the GA in the search for the global optima. Purshouse (2003) explains that this happens when multiple identical solutions have similar fitness values, and thus the algorithm tends to converge prematurely towards one of them. Premature convergence can be a significant issue in a GA effectiveness. The mutation rate can to some extent counteract this effect when promoting diversity in the population, but it must remain between certain low boundaries to avoid converting the evolution process into a simple random search. The undesired phenomenon of genetic drift can be additionally counteracted using mechanisms of niche formation (Purhouse, 2003), such as proposed by Goldberg (1989). Niche formation intends to reduce the selection pressure in densely populated areas of the search space, while promoting the less dense areas (Adra, 2007). Two classic methods are *sharing* and *crowding*. In the crowding approach one parent solution that is very similar to a child solution is replaced deterministically by this offspring by the reinsertion operator. This scheme is simple but not the most efficient. Sharing was proposed by Goldberg & Richardson (1987). The main idea behind sharing is to degrade the fitness of similar solutions. Sharing is further discussed in Appendix A.

B.2. CONSTRAINT HANDLING

Real-world optimization problems involve constraints to be met. They may be constraints in the decision variables, in the values of the objectives to optimize or both. It was mentioned above that the imposition of constraints to an optimization problem could be done either as implicit constraints (a vector of imposed constraints on the objective space) or explicit constraints (limiting directly the feasible values of the decision variables). This gives rise to several constraint handling strategies. Konak et al. (2006) mentions four:

- Forcing the genetic operators to consistently produce feasible solutions. This is mainly based on the coding of the chromosomes, limiting the values of the decision variables to only feasible values (explicit constraints). However, if still infeasible solutions are produced, a more sophisticated approach must be used: Decoders. This enables the chromosome to “give instructions” about how to build feasible solutions (Coello-Coello, 2002)

- Simply discarding infeasible solutions (the “death penalty”). Some authors consider this approach as a penalization (Coello-Coello, 2002) where the fitness of the individual is reduced to zero.
- Using a penalty function to reduce the fitness of infeasible solutions. This is a popular solution. A penalty function reduces the fitness of the individual based on its distance from being feasible (Purshouse, 2003).
- Repairing feasible solutions to transform them into feasible ones. This needs algorithms to transform the infeasible solution into feasible. The fitness of the solution would be reduced in relation to the extent of repair applied.

A fifth approach would be to transform the constraints into additional objectives of the optimization process, and mentioned by Purshouse (2003). A full comprehensive survey of constraint handling techniques can be found in Coello-Coello (2002).

B.3. MULTI-CRITERIA DECISION-MAKING

The process of solving a multi-objective problem is complete only when the process of choosing the right solutions is completed. In real-life problems this is usually to pick just one single solution. This is the task of Multi-Criteria Decision Making (MCDM). This is done by expressing preferences that allows guidance of the decision making process towards the choice of the solution that satisfies the requirements. As Coello-Coello (2000) cites, the expression of preferences can be made handled in three ways:

- *A priori*. The DM has to express the preferences previous to starting the search. This is what is made in the aggregating methods such as the weighted sum approach.
- *A posteriori*. The DM defines the preferences once the search is over and a pool of potential solutions (i.e. the Pareto-optimal set) has been obtained as a result. This is perhaps the most common approach in multi-objective Evolutionary optimization.
- *Interactively*. The DM articulates preferences that guide the search process. Both processes, the definition of preferences interacts with the search process changing over time as the search progresses. This is an ongoing subject of research with several propositions being explored.

The subject of MCDM in evolutionary optimization is a vast field of research being just recently explored. A comprehensive survey of this field can be found in Coello-Coello (2000) and Rachmawati & Srinivasan (2006).

B.4. ADVANCED TOPICS IN THE FONSECA & FLEMING MOGA

B.4.1. Sharing

Sharing was proposed by Goldberg & Richardson (1987). It contributes to counteract the undesired phenomenon of genetic drift (see Section 2.4.4.4). The main idea behind sharing is to degrade the fitness of similar solutions. Through fitness sharing, several individuals close to each other in the objective space have to share resources (i.e. fitness). When sharing the fitness, similar individuals are practically assigned less probability of reproduction, promoting the choice of more diverse characteristics. In other words, similar individuals diminish each other's fitness, and thus their reproducibility. This favours diversification by apportioning the possibility of reproduction to isolated individuals that are equally fit as others. A sharing function is used for estimation of the number of solutions belonging to an optimum:

$$Sh(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}} \right)^\alpha & \text{if } d \leq \sigma_{share} \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.1})$$

The parameter d represents the distance between two solutions. The parameter σ_{share} is called the *niche size*, and it determines the boundaries of the niche within which the residing individuals would share the available resources (i.e. fitness). The normalized distance between any two solutions i and j in a rank can be calculated (given by Deb (2001)) by:

$$d_{ij} = \sqrt{\sum_{k=1}^M \left(\frac{f_k^{(i)} - f_k^{(j)}}{f_k^{max} - f_k^{min}} \right)^2} \quad (\text{B.2})$$

Where f_k^{max} and f_k^{min} are the maximum and minimum objectives function values of the k^{th} objective (for every $k=1, \dots, M$). As could be inferred from Eq. (B.1), if two solutions are far apart a distance $d \leq \sigma_{share}$ they are considered diverse enough not to have sharing effect on each other. In the other case they will have some effect. The parameter α shapes the function; e.g. $\alpha=1$ the effect of sharing reduces linearly from 1 to 0. Each i^{th} individual in the population would have a niche count given by:

$$nc_i = \sum_{j=1}^N Sh(d_{ij}) \quad (\text{B.3})$$

The niche count nc_i is a number ≥ 1 . The parameter d_{ij} is the distance between the i^{th} and j^{th} solutions. The fitness of the individual is then recalculated by:

$$f_i' = \frac{f_i}{nc_i} \quad (\text{B.4})$$

Where f_i is the original fitness value. This means that the fitness of the i^{th} solution is reduced by the other solutions residing in the neighbourhood defined by σ_{share} (Purshouse, 2003). The effectiveness of fitness sharing heavily resides on the correct choice of the niche size σ_{share} . Different GAs implements distinct technique to estimate it. This will be discussed in the section for the MOGA. Niching is usually made in the objective space.

B.4.2. Niching by fitness sharing

MOGA has the specific characteristics of implementing the fitness sharing in the objective space (which permits application to combinatorial optimization problems, Deb (2001)). The parameter σ_{share} (niche size) must be calculated for individuals of only the same rank, dividing the population up into several niches. The sharing parameter indicates the minimum distance between two individuals that does not affect each other's fitness. A niche count is assigned to each individual, set initially to zero and incremented each time another individual meets the requirement for doing so, established by the sharing function (Fonseca & Fleming, 1993). It is possible to estimate σ_{share} solving Eq. (B.5).

$$N\sigma_{share}^{n-1} - \frac{\prod_{i=1}^n (\Delta_i + \sigma_{share}) - \prod_{i=1}^n \Delta_i}{\sigma_{share}} = 0 \quad (B.5)$$

Where $\Delta_i = M - m_i$, and M and m are the two points that define the parallelogram of the objective space, N is the number of individuals, and n is the number of objectives. If there are objectives with different priorities, depending on the compliance of goals in the highest priorities, the sharing parameter can be calculated for only the remaining objectives in trade-off.

Fonseca & Fleming (1995b) later proposed to simplify sharing by computing the niche count following statistical kernel density estimation methods (where the kernel function would become the sharing function). In this way, the niche count and sharing parameter may be based on the Epanechnikov kernel computation.

B.4.3. Mating restriction

The mating of individuals from different far apart individuals may create offspring known as *lethals*: Weak individuals with low fitness that diminish the efficiency of the search process. To discourage the formation of lethals, mating restriction is introduced based on the distance between selected parents. This promotes the mating of genotypically similar individuals (those close to each other in the Pareto-optimal set) in order to create stable niches. The main issue introduced here is the selection of the parameter σ_{mate} which indicates the proximity of

individuals to one another be able to mate. The parameter σ_{mate} may be computed the same way as σ_{share} . In practice, it is sometimes given the same value.

Mating restriction is implemented as follows: After execution of selection, one individual is chosen, and then another one is picked up whose distance is σ_{mate} and then mated. If such an individual does not exist another one is randomly chosen. The size of the population must be big enough to promote the creation of stable niches, according to the number of decision variables. Additionally, coding must be tailored conveniently to the application. The use of Gray codes may be recommendable for its adjacency property (Fonseca & Fleming, 1993).

B.4.4. Articulation of preferences

Fonseca & Fleming (1993) propose the multi-objective genetic optimizer as being composed of a Decision Maker (DM) plus the proper GA. Thus, it is seen as comprising two main blocks: the DM which based on preferences executes the fitness assignment process, and the Genetic Algorithm itself which takes responsibility for the search process (Fig. B.1). The suggested DM permits the progressive articulation of preferences, initially defined in terms of priorities and goals. It is applicable when the goals or priorities (but no necessarily strict preferences) of the optimization can be formulated previous to the search process.

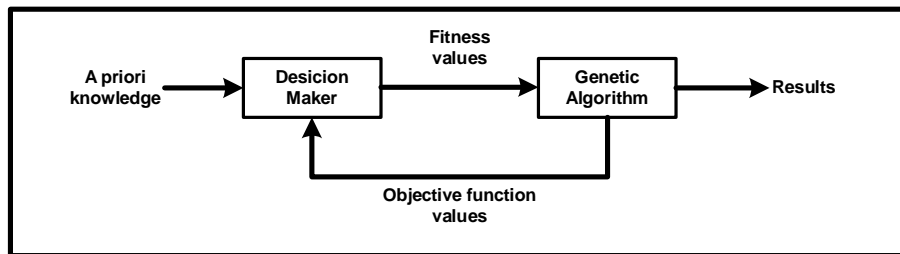


Figure B.1. The General Multi-objective Optimizer proposed by Fonseca & Fleming (1993)

The DM undertakes the strategy for assignment of utility value (ranking), which in turn modifies the fitness allocation. In their first formulation (Fonseca & Fleming, 1993), the approach is based on the goal attainment method. It permits the DM to guide the search towards specific regions of the Pareto set by updating the formulation of goals on each iteration, and comparing different individuals based on their compliance with the goals and their relative non-dominance. This implements the progressive articulation of preferences. In this way, the decision maker influences the way fitness is allocated and thus the entire evolution process.

A further development of the DM strategy was made in Fonseca (1995) and Fonseca & Fleming (1995a, 1998), where a relational preference operator represents the preferences of the DM. The

decision maker structures its choices based on a preferability vector, which defines a utility function based on priorities and goal values. The preferability operator is “*based on Pareto dominance, but it selectively excludes objectives according to their priority and to whether or not they meet their goals*” (Fonseca & Fleming, 1995b). The implementation of iterative progressive articulation of preferences permits the focus of the optimiser’s efforts in a trade-off zone where desired solutions are likely to be, and the DM is provided with this information to refine the preferences, and thus the requirements, of the search.

The comparison operator compares two individuals \mathbf{u} and \mathbf{v} “*in terms of their components with the highest priority, disregarding those in which \mathbf{u} meets the corresponding goals... [if some components of both individuals] meet all goals with this priority, or if they violate some or all of them, but in exactly the same way, the next priority level is considered. The process continues until the lowest priority (1) is reached and satisfied, in which case the result is decided by comparing the priority 1 components of the two [individuals] in a Pareto fashion*” (Fonseca & Fleming, 1995a, 1998). Particular cases are outlined for facilitating the decision making process for cases where the comparison is less straightforward. Once comparison by priorities is exhausted the comparison process (preferability operator) is made such that individual \mathbf{u} is considered preferable over \mathbf{v} if one condition of the following is met:

1. *The violating components of \mathbf{u} dominate the corresponding components of \mathbf{v} .*
2. *The violating components of \mathbf{u} are equal to the corresponding components of \mathbf{v} , but \mathbf{v} violates at least another goal.*
3. *The violating components of \mathbf{u} are equal to the corresponding components of \mathbf{v} , but \mathbf{u} dominates \mathbf{v} as a whole.* (Fonseca & Fleming, 1995b).

Fonseca & Fleming (1995a, 1998) modified the ranking scheme originally proposed. In the new context of the DM, the ranking is now made:

$$\text{rank}(x_u, t) = r_u^{(t)} \quad (\text{B.6})$$

Where x_u is an individual at generation t with the corresponding objective vector \mathbf{u} , and $r_u^{(t)}$ is the number of individuals that are preferable to it. It ranks the individuals in terms of goals and priorities, giving the preferred ones the lower rank. This ensures the preferred individuals are ranked zero. Under this new scheme, not all the non-dominated solutions have the same ranking, but the rank is given in terms of compliance with the order goal-priority.

B.5. DETAILED PROCEDURES IN THE NSGA-II

B.5.1. Non-dominated sorting

The NSGA-II is fully explained in Deb et al. (2000, 2002). Non-dominated sorting determines for each member of the population to which dominance front F_i they belong to. It then assigns the individual's rank according to the belonging front (i.e. non-domination level). In broad terms, the non-dominated set of the population is identified and made front F_1 ($rank=1$). It is then disregarded from the population. A new non-dominated set is identified and assigned front F_2 ($rank=2$). Then this set is also disregarded. The operation continues iteratively until all individuals are assigned to a front F_i and ranked (Deb, 2001).

The procedure by which the algorithm is implemented (Deb et al., 2002), goes as follows:

1. The algorithm assigns each member p a dominance count n_p equal to the number of individuals by which they are dominated, and identified a set S_p of solutions that p dominates.
2. It then finds the non-dominated members (those with $n_p=0$) and allocates them into the first front F_1 with $rank=1$.
3. For each individual q in the front F_1 , the dominance count of the members of its set S_p are reduced by one ($n_p=n_p-1$).
4. If the dominance count n_p of any member q , this is assigned to the next Front F_2 (and given a $rank=2$).
5. This procedure continues iteratively until all members of the population are assigned a front F_i .

B.5.2. Crowding distance density estimation

The crowding distance d_i , is a measure of population density around a particular solution i . It is an estimate of the parameter of the cuboid formed by the vertices fixed by the nearest neighbours (see Fig. B.2). The measure d_i represents the space not occupied by any other solution around i (Deb, 2001). The algorithm sorts the population members in ascending order of magnitude for each of the objectives m . Solutions in the boundaries (for each objective) are assigned an infinite distance value ∞ . For the other individuals, calculate iteratively the distance:

$$d_m^i = d_{m-1}^i + \frac{f_m^{(i+1)} - f_m^{(i-1)}}{f_m^{\max} - f_m^{\min}} \quad \text{for all } m = 1, \dots, M \quad (\text{B.7})$$

Where f_m^{max} and f_m^{min} are the maximum and minimum limits of the m^{th} objective function. The $f_m^{(i+1)} - f_m^{(i-1)}$ is the difference of objective function values of the two neighbouring individuals. The crowding distance d_i represents a measure of population density around i . Therefore, the smaller d_i the more crowded its surrounding environment.

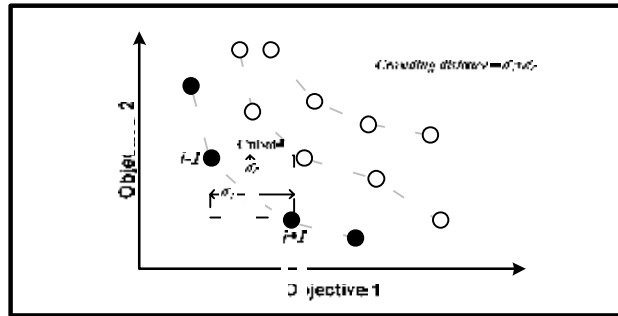


Figure B.2. Crowding distance calculation

B.5.3. Controlled elitism

Emphasis over elite solutions is made twice by NSGA-II: one in the tournament selection and second in selection for survival. This can result in solutions not belonging to non-elitist fronts being deleted too soon. This compromises the balance of exploitation versus exploration. The crowding tournament selection will promote diversity along the non-dominated front. However, *lateral diversity* will be lost. This is the diversity lateral to the non-dominated fronts. This was noticed by Deb and Goel (2001). In order to preserve the good balance, it is necessary to ensure good diversity both along the Pareto-optimal front and also lateral to the front (see Fig. 5.15 in Chapter 5).

Deb and Goel (2001) conceived a mechanism to maintain lateral diversity. This is made by restricting "the number of individuals in the current best non-dominated front adaptively", allowing individuals of diverse non-dominated fronts to be integrated into the new population. The maximum number of individuals of the i^{th} front ($i=1,2,\dots, K$) N_i permitted in the new population is restricted by:

$$n_i = N_{pop} \frac{1-r}{1-r^K} r^{i-1} \quad (\text{B.8})$$

Where N_{pop} is the size of the population and r is the reduction rate ($r < 1$). With this, "each front is allowed an exponentially reducing number of solutions". Notice that the allowable number of individuals from the first front is the highest, and it decreases progressively with each front. Nevertheless there may be a number of individuals in the front i different from n_i : $n_i^t \neq n_i$. This is

solved by the procedure described below. Figure B.3 shows the flowchart of the controlled elitism mechanism. Also note that Figure 5.13 (in Chapter 5) indicates which part of the original NSGA-II algorithm would be replaced with the controlled elitism mechanism.

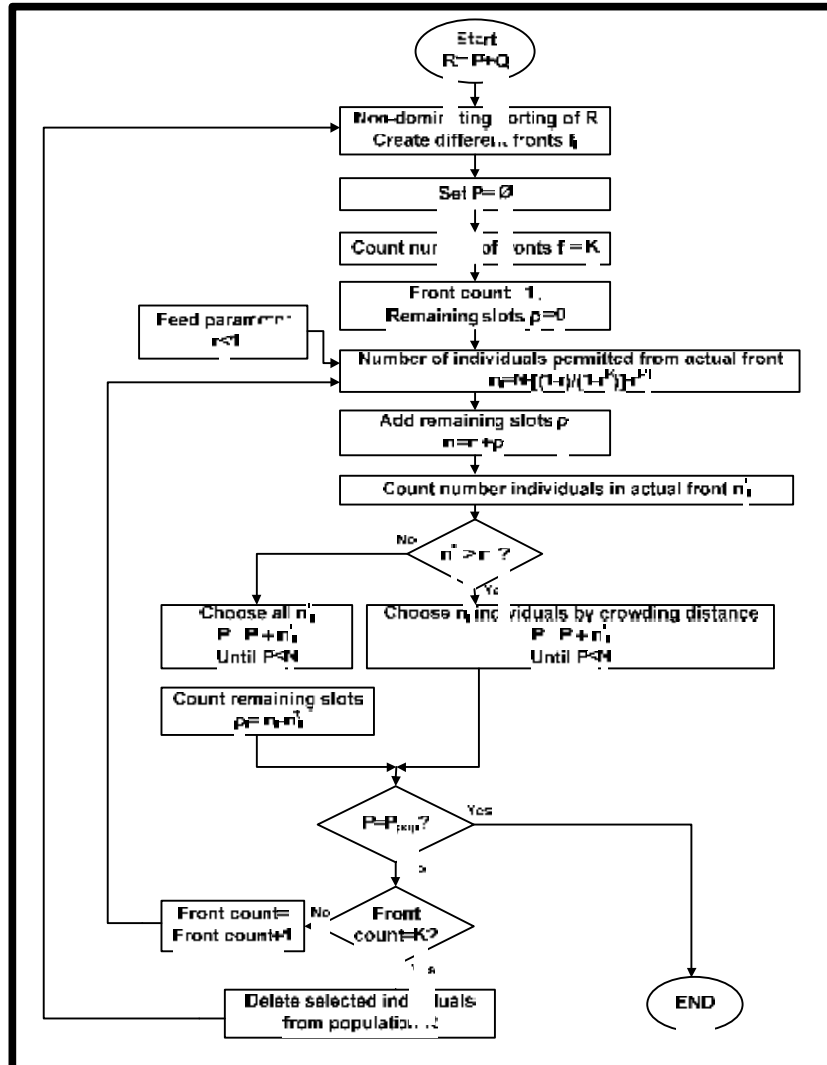


Figure B.3. Controlled Elitism Algorithm

The procedure that implements the controlled elitism goes as follows:

1. The population $\mathbf{R}=\mathbf{P}+\mathbf{Q}$ is non-dominated sorted. Set $\mathbf{P}=\emptyset$. Count number of fronts $F_i=1,2,\dots,K$.
2. Calculate the number of individuals permitted for the actual front n_i .
3. Count the number of individuals in the first front n_i^f .

4. If $n_1^t > n_1$ (there are more individuals than those allowed), then insert only n_1 individuals into the new population (choosing them by crowded tournament selection; $\mathbf{P}_{t+1} = \mathbf{P}_t + n_1$). If on the contrary $n_1^t < n_1$ (there are less solutions than needed), then insert all solutions n_1^t ($\mathbf{P}_{t+1} = \mathbf{P}_t + n_1^t$) and make a count of the remaining slots $\rho_1 = n_1 - n_1^t$.
5. If $\mathbf{P} = \mathbf{N}_{\text{pop}}$, end the procedure. If $\mathbf{P} < \mathbf{N}_{\text{pop}}$, go back to step 2. Calculate the number of allowed individuals in the second front n_2 . Add to it the number of remaining slots: $n_2 = n_2 + \rho_1$. Compare n_2^t against n_2 in the same fashion. Repeat iteratively the procedure with the remaining fronts until $\mathbf{P} = \mathbf{N}_{\text{pop}}$.
6. If the case were that after exhausting all K fronts still $\mathbf{P} < \mathbf{N}_{\text{pop}}$, then go over all the procedure again from step 1.

Notice that the parameter r is problem-dependent and thus an additional parameter to tune.